VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Automatický systém měření signálové úrovně v mobilní síti

Automatic System of Signal Strength Measurement in Mobile Network

2017 Martin Talaš

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání diplomové práce

Student: **Bc. Martin Talaš**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2601T013 Telekomunikační technika

Téma: Automatický systém měření signálové úrovně v mobilní síti
Automatic System of Signal Strength Measurement in Mobile Network

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem diplomové práce je návrh a implementace systému pro měření kvalitativních parametrů sítě mobilního operátora.

1. Popis technologií mobilních sítí.
2. Šíření signálu mobilní sítě a měření signálové úrovně.
3. Návrh systému automatického měření RSSI pro vozidla taxi služby.
4. Realizace klientské části s mobilním terminálem, GPS a WiFi AP.
5. Implementace serverové části s logováním dat a vizualizací v GIS.
6. Zhodnocení dosažených výsledků.

Seznam doporučené odborné literatury:

[1] S. Sesia, I. Toufik, M. Baker, LTE - The UMTS Long Term Evolution: From Theory to Practice, Wiley, 2011, 792 p., ISBN 978-0470660256.
[2] J. Eberspächer, H. Vögel, C. Bettstetter, C. Hartmann, GSM - Architecture, Protocols and Services, Willey, 2009, 338 p., ISBN 978-0470030707.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **doc. Ing. Miroslav Vozňák, Ph.D.**

Datum zadání: 1.9.2016

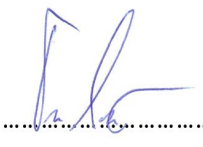
Datum odevzdání: 28.4.2017

doc. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 27. dubna 2017 

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 27. dubna 2017 ……

Rád bych na tomto místě poděkoval doc. Ing. Miroslavu Vozňákovi, Ph.D. za vedení této práce a jeho věcné rady, připomínky a konzultace, kolegovi Markovi Zahradníkovi za konzultace při řešení problematiky Pythonu a za zapůjčení serverových kapacit, Ludovítu Barančekovi za pomoc při práci s PHP, kolegovi Miloslavu Hajnému za dny volna v práci a za všudypřítomný pozitivní přístup, kamarádům Jakubu Martiníkovi, Martinu Brožovičovi a Jiřímu Dvořákovi za podporu, Karolíně Vackové za pevné nervy a rodině za jejich trpělivost.

Abstrakt

Tato diplomová práce se zabývá návrhem a realizací systému automatického měření parametrů signálu mobilní sítě, který je postaven na jednodeskovém počítači RaspberryPi s LTE modemem, GPS příjímačem a WiFi AP. Systém je topologie multi-klient server a pracovní prostředí tvoří operační systém Linux. Serverová část zajišťuje sběr dat z částí klientských a prezentuje tato data na mapě ve webovém rozhraní. Systém je možné použít při mapování parametrů mobilní sítě za účelem její optimalizace.

Klíčová slova: RSSI, RSRP, SINR, RSRQ, RSCP, Ec/No, LTE, GSM, UMTS, WCDMA, měření signálu, Linux, RaspberryPi, PHP, Python, šíření signálu, Raspbian, WiFi, AP, úroveň signálu

Abstract

This diploma thesis deals with design and implementation of automatic system measuring qualitative signal parameters based on single-board computer RaspberryPi with LTE modem, GPS receiver and WiFi Access Point. The system is multi-client server topology based purely on Linux operating system. The server collects data from clients and visualize them on map running on its web page. The whole system can be used for mapping of mobile signal strength and quality, and may help mobile operators with overall network optimization.

Key Words: RSSI, RSRP, SINR, RSRQ, RSCP, Ec/No, LTE, GSM, UMTS, WCDMA, signal measurement, Linux, RaspberryPi, PHP, Python, signal propagation, Raspbian, WiFi, AP, signal level

Obsah

Seznam použitých zkratek a symbolů

3GPP	–	3rd Generation Parthership Project
AAA	–	Authentication/Authorization/Access
AN	–	Access Network
AP	–	Access Point
API	–	Application Programming Interface
AuC	–	Authentication Centre
BCCH	–	Broadcast Channel
BER	–	Bit Error Rate
BPSK	–	Binary Phase Shift Keying
BSC	–	Base Station Controller
BSS	–	Base Station Subsystem
BTS	–	Base Transceiver Station
CDMA	–	Code Division Multiple Access
CN	–	Core Network
CPICH	–	Common Pilot Channel
CQI	–	Channel Quality Indicator
CS	–	Circuit Switched
DC-HSPA	–	Dual Carrier High Speed Packet Access
E-UTRAN	–	Evolved Universal Terrestrial Radio Access Network
EARFCN	–	EUTRA Absolute Radio Frequency Channel Number
Ec/No	–	Energy of Chip devided by Noise power density
EDGE	–	Enhanced Data Rates for GSM Evolution
EIR	–	Equipment Identification Register
EPC	–	Evolved Packet Core
ETSI	–	European Telecommunications Standards Institute
FDD	–	Frequeny Division Duplex
FDMA	–	Frequency Division Multiple Access
GGSN	–	Gateway GPRS Support Node
GLONASS	–	Global Navigation Satellite System
GMSC	–	Gateway Mobile Switching Centre
GPRS	–	General Packet Radio Service
GPS	–	Global Positioning System
GSM	–	Global System for Mobile Communication
HLR	–	Home Location Register
HSPA	–	High Speed Packet Access
HSS	–	Home Subscriber Server

HTML	–	HyperText Markup Language
HTTP	–	Hypertext Transfer Protocol
IP	–	Internet Protocol
ITU	–	International Telegraph Union
ITU-T	–	ITU Telecommunication Standardization Sector
LAC	–	Location Area Code
LTE	–	Long Term Evolution
LTE-A	–	Long Term Evolution Advanced
MME	–	Mobile Management Entity
MMSC	–	Multimedia Messaging Service Centre
MOS	–	Mean Opinion Score
MS	–	Mobile Station
MSC	–	Mobile Switching Centre
NAT	–	Network Address Translation
NMEA	–	National Marine Electronics Association
NMT	–	Nordic Mobile Telephone
NSS	–	Network Subsystem
O&M	–	Operations and Maintenance
OFDM	–	Orthogonal Frequency Division Multiplex
OFDMA	–	Orthogonal Frequency Division Multiple Access
P-GW	–	Packet Data Network Gateway
PDN	–	Packet Data Network
PESQ	–	Perceptual Evaluation of Speech Quality
PRB	–	Physical Resource Block
PS	–	Packet Switched
QAM	–	Quadrature Amplitude Modulation
QPSK	–	Quadrature Phase Shift Keying
QZSS	–	Quasi-Zenith Satellite System
RNC	–	Radio Network Controller
RNS	–	Radio Network Subsystem
RSCP	–	Received Signal Code Power
RSRP	–	Reference Signal Receive Power
RSRQ	–	Reference Signal Receive Quality
RSSI	–	Reference Signal Strength Indicator
S-GW	–	Serving Gateway
SC-FDMA	–	Single Carrier - Frequency Division Multiple Access
SF	–	Spreading Factor
SGSN	–	Serving GPRS Support Node
SINR	–	Signal-to-Interference and Noise Ratio

SMSC	–	Short Message Service Centre
SS7	–	Signalling System No. 7
TDD	–	Time Division Duplex
TDMA	–	Time Division Multiple Access
UE	–	User Equipment
UMTS	–	Universal Mobile Telecommunications System
URL	–	Uniform Resource Locator
UTRAN	–	Universal Terrestrial Radio Access Network
VLR	–	Visitor Location Register
WCDMA	–	Wideband Code Division Multiple Access

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

1 Úvod

S příchodem mobilních sítí a jejich prvním nasazením vznikaly požadavky na kvalitu přenášeného signálu a jeho měření. I když jsou dnes sítě první generace minulostí, tyto požadavky přetrvávají dodnes a hrají hlavní roli při výstavbě a optimalizaci mobilní sítě.

První čast této práce se zaměřuje na popis a základní seznámení s nejvíce rozšířenými mobilními sítěmi dnešní doby, jako jsou sítě GSM, UMTS a LTE. Popis má dát jasný přehled o základní struktuře každé sítě, a také jejich přístupové části. Ta hraje velmi důležitou roli pro pochopení jednolivých měřených parametrů.

Ve druhé části práce je popsáno šíření mobilního signálu prostorem a parametry mající vliv na jeho výslednou přijímanou úroveň, jako jsou útlum trasy, útlum stínění či vícecestné úniky. Dále jsou popsány kvalitativní parametry signálu každé ze zmiňovaných sítí, jejich význam a odpovídající jednotky.

Poslední část práce se zabývá návrhem a realizací automatického systému pro měření úrovně signálu. Hlavní myšlenkou je implementace měřících zařízení do vozidel taxi služby za účelem plošného mapování parametrů signálu. V sekci návrhu je popsán princip funkce a základní požadavky na měřící systém, především pak na měřené parametry. Následně je podrobně popsána realizace klientské části postavené na jednodeskovém počítači RaspberryPi a serverové časti shromažďující data za účelem vizualizace na mapovém podkladu.

2 Technologie mobilních sítí

Historie bezdrátové komunikace se datuje přibližně od 90. let 19.století, kdy americký zubař Mahlon Loomis demonstroval přenos telegrafního signálu přes dvojci vzdušných draků na vzdálenost 22 kilometrů a kdy vznikají první experimenty s přenosem signálu radiovými vlnami svého druhu obecně [1]. Postupem času se technologie zdokonalovaly, a to především kvůli stále se zvyšujícím požadavkům na kvalitu, přenosové zpoždění, propustnost a mobilitu.

Dnešní mobilní sítě jsou ve většině z případů složeny z více technologií mající ať už svůj historický význam, tak zajišťující relevantní konkurenceshopnost. Je to dáno především faktem, že novější technologie pracují na odlišných principech, se kterými velká část starších, stále aktivních mobilních zařízení není zcela kompatibilní.

Mezi stále aktivní buňkové mobilní sítě patří především sítě 2. generace GSM (Global System for Mobile Communication), 3. generace CDMA (Code Division Multiple Access) a UTMS (Universal Mobile Telecommunications System) a 3,5. až 4. generace LTE (Long Term Evolution) a LTE-A (Long Term Evolution Advanced). Každá z těchto technologií je popsána svou topologií, jednotlivými síťovými prvky a typem síťového přístupu, tzv. *"Access Network"* (AN). Vzájemné propojení sítí je pak založeno buď na přepínání okruhů *"Circuit Switched"* (CS), nebo přepínání paketů *"Packet Switched"* (PS).

2.1 Technologie GSM

Technologie GSM patří do sítí 2. generace a byla vyvinuta na konci 90. let 20. století. Poprvé byla nasazena ve Finsku koncem roku 1991. [1] GSM je postavena na přepínání okruhů. Pro svou signalizaci využívá sadu protokolů SS7 a pracuje v rozsahu frekvenčních pásem 710 - 900 MHz a 1800 - 1900 MHz.

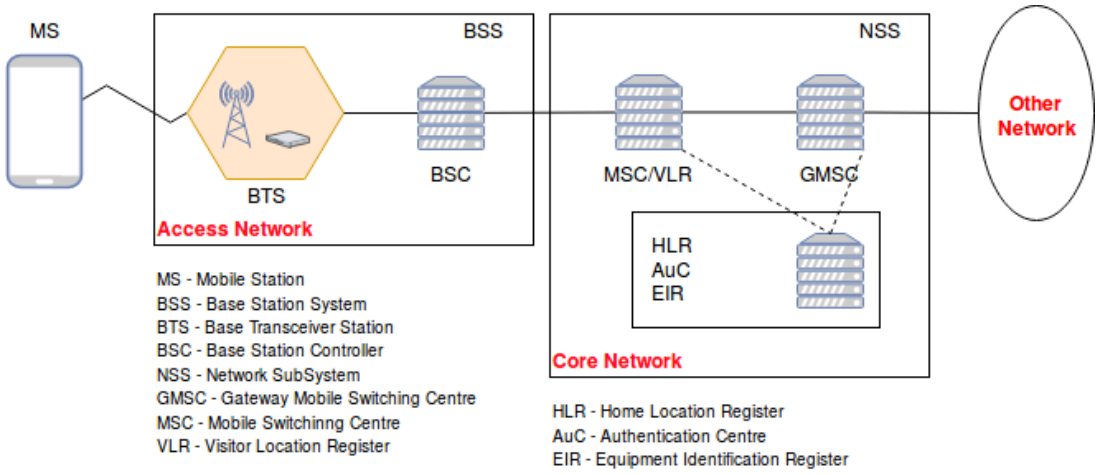
GSM není pripárně určena na přenos aplikačních dat. Pro tyto služby byla rozšířena o technologie GPRS a EDGE, jejichž popis není náplní této práce. Jsou však využívány v praktické části pro přenos dat z klienta na server v případě, že je klient registrován v sítí GSM.

2.1.1 Přístupová síť

GSM technologie využívá na své přístupové síti kombinaci časového (TDMA) a frekvenčního (FDMA) multiplexu. V praxi to znamená, že dostupné frekvenční pásmo je rozděleno na kanály o šířce 200 kHz (s odpovídajícími ochrannými pásmy), kde každý takovýto kanál obsahuje 8 TDMA slotů. Tyto timesloty jsou obsluhovány periodicky s délkou trvání 577us. Z výše uvedeného vyplývá, že datový přenos mezi účastníkem a základnovou stanicí musí být časově synchronní. [3]

2.1.2 Struktura sítě GSM

GSM síť je pomyslně rozdělena na přístupovou síť a tzv. "Core Network"(CN). Přístupová síť vytváří prostředí pro připojení mobilních uživatelů a její součástí jsou rádiové stanice BTS a jejich kontrolery BSC. Součástí CN jsou pak prvky MSC řídící provoz mezi MS a CN, HLR/VLR uchovávající mimo jiné aktuální pozici MS v síti, AuC zajišťující autentizaci MS, pomocný registr EIR kontrolující validitu mobilního zařízení a GMSC zajišťující konektivitu s vnější sítí. Celé schéma je zobrazeno na obrázku 1.



Obrázek 1: Základní struktura GSM sítě

Nutno podotknout, že rozšířená GSM síť může obsahovat více prvků v závislosti na poskytovaných službách (SMSC, MMSC, apod.).

2.2 Technologie CDMA

Mobilní síť postavená na technologii CDMA, přesněji CDMA2000 1xEV-DO Rev.A, patří především v České Republice k uživatelsky méně využívaným. Je to dáno především typem své přístupové sítě, ale také použitými frekvenčními pásmy 410 MHz a 450 MHz. Ty byly v minulosti využívány 1. generací mobilních sítí NMT. Obrovskou výhodou této sítě je právě její použité frekvenční pásmo, kterým se signál při stejném vysílacím výkonu šíří na přibližně dvojnásobnou vzdálenost než v případě GSM frekvence 900 MHz. Takto se dá efektivně pokrýt velké území s relativně malým množstvím základnových stanic.

Přístupová síť technologie CDMA je založena na tzv. kódovém multiplexu. Jedná se o metodu přístupu ke sdílenému přenosovému médiu (přenosovému kanálu), kde každý uživatel v dané oblasti může komunikovat ve stejný okamžik na stejné frekvenci. Aby se zabránilo vzájemným interferencím, každý uživatel navíc násobí data přiděleným signálem rozprostřeného spektra, tzv. *"Spreading signal"*. Tento proces se v angličtině nazývá *"spreading"* a výstupem je signál rozprostřený po celé šířce přenosového pásma. Jelikož jsou jednotlivé signály rozprostřeného

spektra ortogonální, základnová stanice nemá problém signály od jednotlivých uživatelů odlišit. [2] [3]

Nevýhodou této sítě je relativně malá propustnost, která v případě CDMA2000 1xEV-DO Rev.A činí teoreticky 3,1 Mbps na "downlink"a 1,8 Mbps na "uplink". Tato síť bude v České Republice v budoucnu pravděpodobně nahrazena sítí LTE a při řešení této diplomové práce se jí více nebudeme zabývat.

2.3 Technologie UMTS

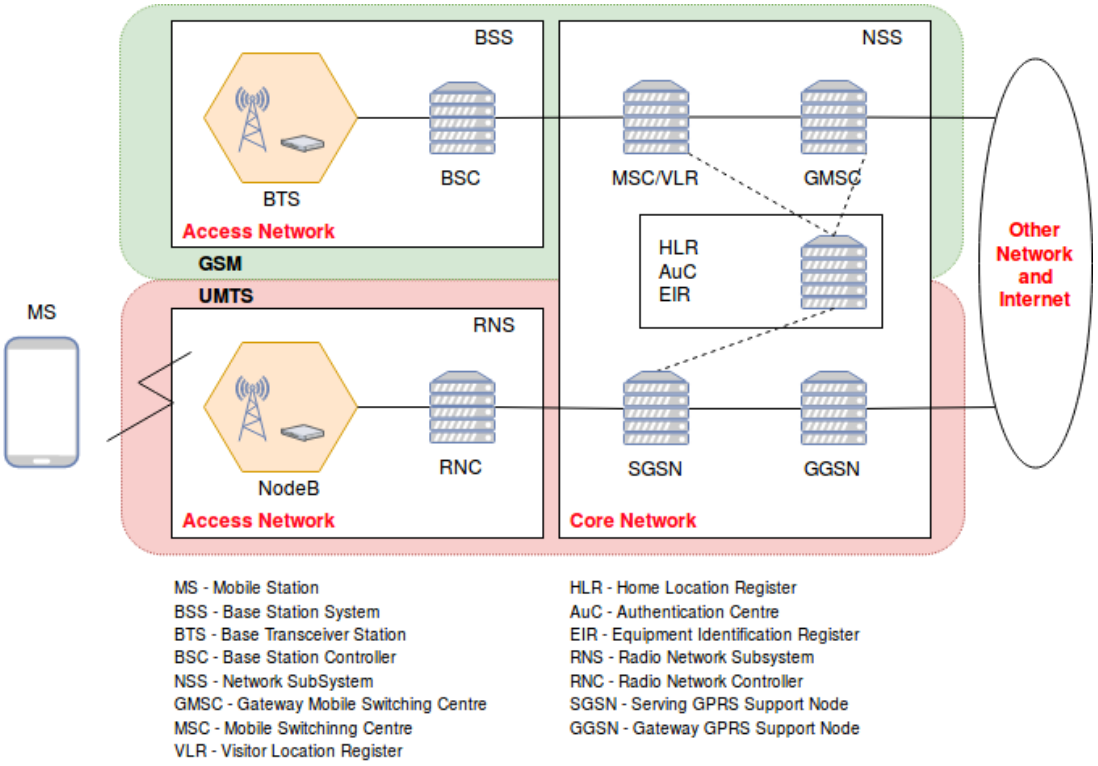
UMTS je částečně paketově a částečne okruhově přepínaná síť přinášející oproti GSM především vyšší propustnost a lepší konektivitu pro přenos aplikačních dat. Frekvenční pásma UMTS jsou definována v rozmezí 700 MHz - 3500 MHz, nejčasněji se však využívá pásmo 2100 MHz.

2.3.1 Přístupová síť

Třetí generace mobilních sítí přináší novou přístupovou síť založenou na širokopásmovém kódovém multiplexu WCDMA (Wideband Code Division Multiple Access). Pro modulaci se využívá fázové kláčování QPSK, ve vyšších verzích HSPA pak navíc 16-ti stavová QAM. Výsledkem modulace jsou symboly, které se v závislosti na "Spreading"faktoru (SF) rozdělí na tzv. "chipy", které už přenášejí samotnou informaci k MS. Tento princip vychází z technologie CDMA popsané v sekci 2.2.

2.3.2 Struktura sítě UMTS

Jak bylo řečeno, síť UMTS pracuje jak v paketové, tak v okruhové doméně, kde přepánání okruhů slouží pro hlasové služby a přepínání paketů pro služby datové. Struktura síťě je znázorněna na obrázku 2



Obrázek 2: Základní struktura UMTS sítě

Jak je zřejmé, technologie UMTS je zpětně kompatibilní se sítí GSM a v praxi funguje spíše jako její rozšíření. Novými prvky přístupové sítě jsou základnové stanice NodeB a jejich kontroléry RNC. Z pohledu CN je to pak SGSN zajišťující směrování a správu paketových spojení v síti a GGSN sloužící jako brána do vnějších sítí. V praxi jsou většinou MSC a SGSN součástí jednoho zařízení.

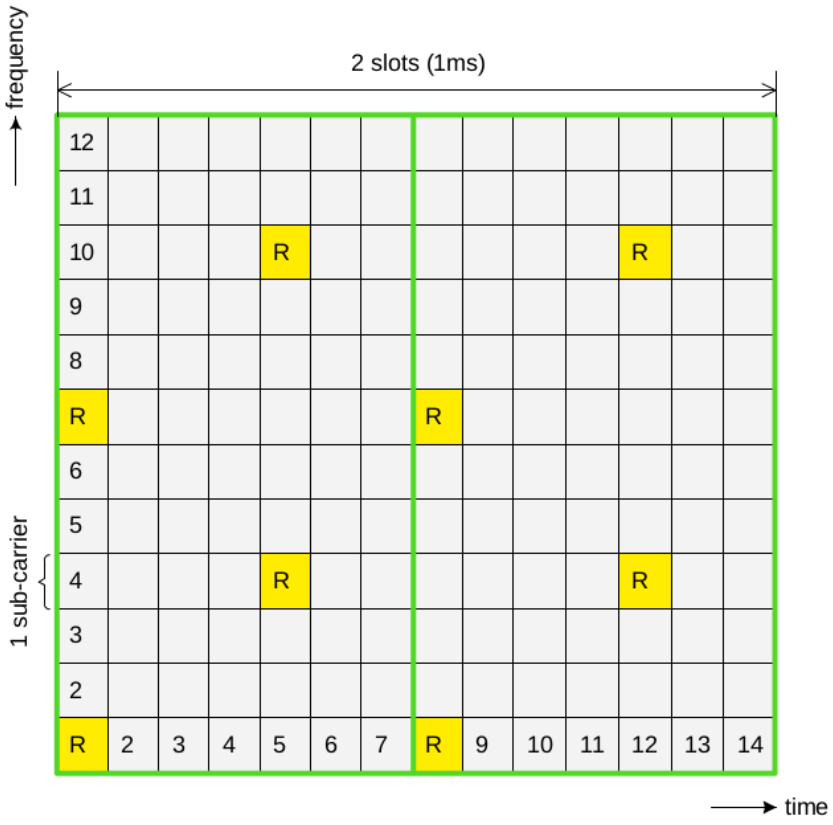
2.4 Technologie LTE/LTE-A

LTE (LTE/EPC) je plně paketově orientovaná síť zajišťující nepřetržítou IP konektivitu mezi MS a paketovou datovou sítí (PDN). Od svého prvního spuštění v roce 2009 je široce nasazována především kvůli přenosovým rychlostem a efektivnímu hospoařření s frekvenčním pasmem. LTE je provozováno jak ve frekvenčním (FDD), tak časovém duplexu (TDD).

2.4.1 Přístupová síť

LTE využívá na fyzické vrstvě své přístupové sítě OFDMA pro “downlink” a SC-FDMA pro “uplink”. Pro modulaci je použita kvadraturní amplitudová modulace 16QAM/64QAM nebo fázové klíčování BPSK/QPSK. Struktura dat je v LTE rozdělena na rámce s délkou trvání 10 ms, které jsou dále rozděleny na 10 subrámců s délkou trvání 1 ms. Každý subrámec obsahuje dva kanálové intervaly, tzv. “timesloty”. Kanálový interval může mít 6 nebo 7 OFDM symbolů v závislosti na délce ochranného intervalu, tzv. “cyklického prefixu”.

Nejmenší blok dat, se kterým lze v průběhu přenosu manipulovat je tzv. PRB *(Physical Resource Block)*. Ten se skládá z dvanácti subrámců, každý s rozdílnou subnosnou frekvencí, jak znázorňuje obrázek 3. Uvnitř PRB jsou systematicky rozprostřeny referenční symboly sloužící mimo jiné pro měření kvalitativních parametrů, jak je popsáno níže v sekci 3.5.

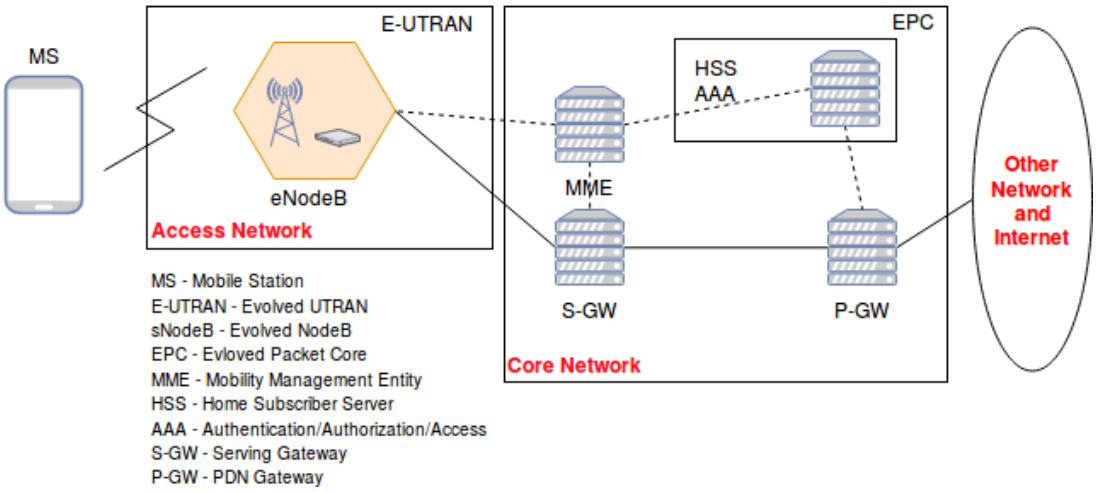


Obrázek 3: Struktura PRB v LTE [4]

Šířka kanálu v LTE může být 1.4, 3, 5, 10, 15 nebo 20 MHz.

2.4.2 Struktura sítě LTE

Přístupovou síť tvoří v LTE základnové stanice eNodeB. Ty už nejsou spravovány pomocnýmu kontroléry jako v případě GSM a UMTS, ale přímo signalizačními prvky MME z CN. Navíc má eNodeB rozhraní X2 sloužící k přímé komunikaci mezi sousedními základnovými stanicemi pro potřeby handoveru a správě toku dat. Základní struktura sítě LTE je znázorněna na obrázku 4.



Obrázek 4: Základní struktura LTE sítě

- **MME** *(Mobility Management Entity)* - kontrolní prvek zajišťující signalizaci mezi MS a CN.
- **HSS** *(Home Subscriber Server)* - obdoba HLR v GSM síti sloužící pro uchování informací o účastníkovi, jeho poloze v síti, IP adrese a dostupných službách.
- **AAA** *(Authentication/Authorization/Access)* - 3GPP AAA server autentizující uživatele v síti.
- **S-GW** *(Serving Gateway)* - prvek zajišťující správu veškerého toku uživatelských IP paketů v síti.
- **P-GW** *(PDN Gateway)* - prvek spravující přidělování IP adres v síti, QoS a konektivitu s vnější sítí.

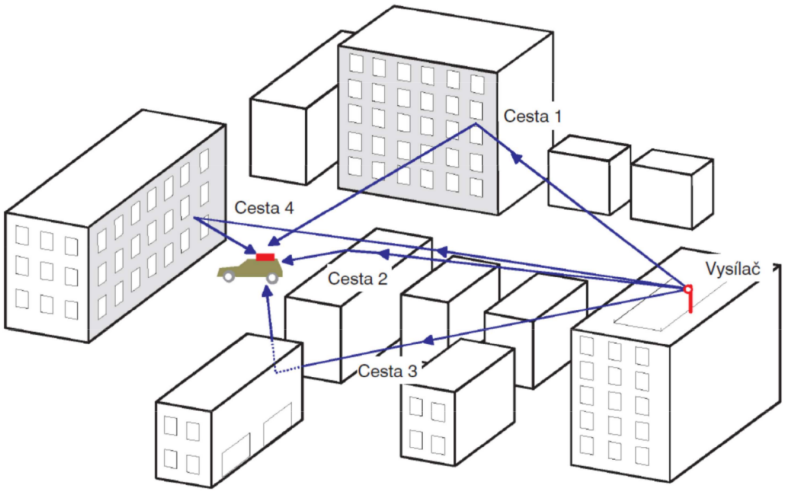
3 Šíření signálu mobilní sítě a měření signálové úrovně

3.1 Úvod

Šířením signálu v mobilní síti rozumíme bezdrátový přenos informace mezi základnovou stanicí a mobilním zařízením. I přes fakt, že každá mobilní technologie pracuje na rozdílných principech kódování, modulace či umožňuje vícenásobné šíření, samotný přenos signálu se v podstatě neliší. Jak pro GSM, tak pro LTE sítě platí základní pravidla šíření elektromagnetických vln, a také se musí potýkat s vícecestným šířením signálu.

3.2 Šíření signálu

Signál se v mobilní síti šíří prostorem jak přímo, tak vícecestně v závislosti na zástavbě, porostu či členitosti terénu. Příklad vícecestného šíření je znázorněn na obr. 5



Obrázek 5: Příklad vícecestného šíření signálu [5]

Velikost přijímacího výkonu P_{Rx} takto šířeného signálu můžeme popsat vztahem

$$P_{Rx} = P_{Tx} \cdot g_{Tx} \cdot g_{Rx} \cdot g_c \quad [dBm], \tag{1}$$

kde P_{Tx} je velikost vysílacího výkonu, g_{Tx} a g_{Rx} jsou zisky vysílací a přijímací antény a g_c je celkový útlum na radiovém kanálu. Ten lze dále rozdělit na útlum vztažený k délce trasy $g_d(L)$, útlum stínění g_s a vícecestné úniky g_m, jak je znázorněno ve vzorci 2 [3].

$$g_c = g_d(L) \cdot g_s \cdot g_m \quad [dB] \tag{2}$$

- **Útlum trasy $g_d(L)$** je obvykle modelován jako deterministická funkce vzdálenosti L mezi vysílačem a přijímačem.

$$g_d(L) = \left(\frac{\lambda}{4\pi L}\right)^2 \cdot \left(\frac{L_0}{L}\right)^{\gamma-2} \sim L^{-\gamma} \quad [dB], \tag{3}$$

kde λ odpovídá vlnové délce, L_0 je referenční vzdálenost získaná z měření v blízkosti vysílače a $\gamma \geq 2$ je útlumový exponent popisující tempo nárustu útlumu v závislosti na prostředí šíření. Typické tabulkové hodnoty γ se pohybují v rozmezí 3-5 v závislosti na zástavbě. [3] [6]

- **Útlum stínění g_s** popisuje efekt kolísání přijímaného výkonu kolem své střední hodnoty a je způsoben překážkami na přenosové trase, jako jsou budovy a vegetace. Útlum stínění je oobecně popsán statistickým modelem s logaritmicko-normálním rozělením náhodné veličiny. Útlum stínění v decibelech

$$\chi = 10log(g_s) \quad [dB] \tag{4}$$

je distribuován v Gaussově rovině vztahem

$$f_\chi(\chi) = \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\chi^2/2\sigma^2}. \tag{5}$$

Standardizovaná odchylka σ určuje ráz stínění a je závislá na prostředí šíření. Typické měřené hodnoty σ se pohybují v rozmezí 5-10 dB. [3]

- **Vícecestné úniky g_m** popisují další zdroj kolísání přijímaného výkonu kolem své střední hodnoty a jsou způsobeny vícecestným šířením signálu. V oblastech s vyšší hustotou zástavby se vysílaný signál odráží a vytváří tak své kopie, které přicházejí do přijímače z různých směrů a s různým zpožděním, viz. obr. 5. Úroveň přijímaného signálu pak odpovída součtu amplitud přijímaných signálů na přijímači v jednotce času. Variace výsledné amplitudy lze vyjádřit vztahem

$$g_m = a^2 \quad [-], \tag{6}$$

kde a je náhodná veličina závisející na prostředí šíření. Pokud neexistuje spojení mezi vysílačem a přijímačem na přímou viditelnost, a odpovídá veličině s Rayleigheho distribucí. g_m je pak distribuována exponenciálně. [3]

3.3 Kvalitativní parametry GSM signálu

3.3.1 Reference Signal Strength Indicator (RSSI)

V GSM síti je všeobecně měřen pouze parametr RSSI udávající přijímanou výkonovou úroveň v rámci dané šířky pásma. Měření je prováděno broadcastovém kanálu BCCH a jednotkou je dBm. [7]

3.4 Kvalitativní parametry UMTS signálu

3.4.1 Reference Signal Code Power (RSCP)

RSCP v UMTS reprezentuje přijímaný výkon na jednom kódovém symbolu *"chip"* měřený na primárním CPICH kanálu. Pokud je na straně vysílače použita diverzitní anténa, RSCP odpovídá součtu samostatně měřených výkonů primárního CPICH kanálu. Jednotkou je dBm. [7]

3.4.2 Reference Signal Strength Indicator (RSSI)

Přijímaná výkonová úroveň spolu s tepelným šumem a šumem generovaným na přijímači, odpovídá hodnotě RSSI v jednotce dBm. [7]

3.4.3 CPICH Ec/No

Parametr Ec/No vyjadřuje přijímanou energii jednoho chipu (Ec) podělenou výkonem šumu a interneferencí. Jelikož jsou ale obě hodnoty v jednotkách dBm, můžeme výslednou hodnotu vyjádřit vztahem [7]

$$Ec/No = RSCP - RSSI \quad [dB]. \tag{7}$$

3.5 Kvalitativní parametry LTE signálu

3.5.1 Reference Signal Receive Power (RSRP)

Parametr RSRP je definován jako průměrná hodnota přijímaného výkonu referenčních symbolů v jednotkách decibel-miliwatt (dBm) přenášejících referenční signály skrz dané frekvenční pásmo. Rozsah měřených hodnot se pohybuje v rozmezí -140 dBm až -44 dBm. Hodnotu lze zformulovat jako vysílací výkon L ponížen o útlum trasy A_p a povýšen o zisk přijímací antény g_{Rx}

$$RSRP = L + g_{Rx} + A_p \quad [dBm]. \tag{8}$$

Celkový vysílací výkon v případě LTE závisí na vysílacím výkonu základnové stanice $P_{Tx}[mW]$ a počtu PRB odpovídající použité šířce pásma.

$$L = 10log\Big(\frac{P_{Tx}}{12 \cdot N_{PRB}}\Big) \quad [dBm] \tag{9}$$

Referenční bod pro měření RSRP je anténní konektor na mobilním zařízení UE *(User Equipment)* [7] [8].

3.5.2 Reference Signal Strength Indicator (RSSI)

Parametr RSSI je indikátor síly přijímaného signálu. Jeho jednotkou je dBm a je definován jako celkový průměr přijímaného výkonu ze všech:

- OFDM symbolů obsahujících referenční symboly pro měření ze všech dostupných zdrojů,
- kanálů všech buněk v dosahu,
- interferencí přilehlých kanálů,
- tepelný šum, apod.

Můžeme ho tedy vyjádřit jako součet přijímaného výkonu, interferencí a šumu. [7]

$$RSSI = S_{tot} + I_{tot} + N_{tot} \quad [dBm] \tag{10}$$

3.5.3 Reference Signal Receive Quality (RSRQ)

RSRQ je parametr reprezentující kvalitu přijímaného signálu je definován vztahem

$$RSRQ = N_{PRB} \cdot \frac{RSRP}{RSSI} \quad [dB], \tag{11}$$

kde N_{PRB} je počet bloků PRB *(Physical Resource Block)*, pro které bylo měření provedeno. Nutno zmínit, že měření RSRP a RSSI by v tomto případě mělo být provedeno na stejné skupině PRB. Referenční bod pro měření RSRQ je opět anténní konektor na UE. [7]

3.5.4 Signal to Interference Noise Ratio (SINR)

I když SINR není definován v doporučeních a směrnicích ETSI, považuje se za kvalitativní parametr LTE signálu a je definován vztahem

$$SINR = \frac{RSRP}{I \cdot N} \quad [dB], \tag{12}$$

kde I odpovídá přijímané výkonové úrovni interferujících signálů a N reprezentuje šum. Výpočet SINR slouží k určení hodnoty CQI *(Channel Quality Indicator)* sloužící ke zpětnému informování základnové stanice o úrovni signálu na UE. Tyto informace slouží především pro potřeby handoveru a ke korekcím přenosových rychlostí.

4 Automatický systém pro měření signálové úrovně

4.1 Návrh měřícího systému

Měřící systém vychází z reálných požadavků na uniformní mobilní zařízení, schopné v jednotkách času zaznamenávat kvalitativní parametry mobilní sítě a spolu s polohu z navigačního systému tato data odesílat na server a zobrazit na mapovém podkladu. Výstupem tohoto systému je webová prezentace zobrazující jak samotnou mapu, tak prvky umožňující práci s daty a jejich následný export.

4.1.1 Požadavky na měřené parametry mobilní sítě

Systém by měl být schopen měřit kvalitativní parametry většiny veřejně dostupných mobilních sítí, jako jsou GSM, UMTS a LTE. Rozsah měřených dat by měl být co největší s ohledem na použité komponenty a celkovou realizaci. Minimálně jsou však stanoveny tyto základní parametry sítě:

- **LTE**
  - RSSI, RSRP, RSRQ, SINR
- **UMTS**
  - RSSI, RSCP, Ec/No
- **GSM**
  - RSSI

4.1.2 Požadavky na měřené parametry z GPS

Hlavním požadavkem na měřící systém je schopnost sběru a vyhodnocování údájů o poloze. Měřící systém by proto měl obsahovat GPS přijímač nebo jiné zařízení schopné získávat údaje o aktuální poloze. Těmito údaji jsou:

- Zeměpisná šířka *(Latitude)*
- Zeměpisná délka *(Longitude)*

4.1.3 Ostatní požadavky

Klientská část by měla být schopna udržovat nepřetržitou komuninakaci se serverem za účelem shromažďování naměřených dat a také propotřeby Q&M. Z tohoto důvodu by měla podoprovat i možnost lokálního přístupu, a to pomocí rozhraní Ethernet nebo bezdrátově vlastním WiFi AP.

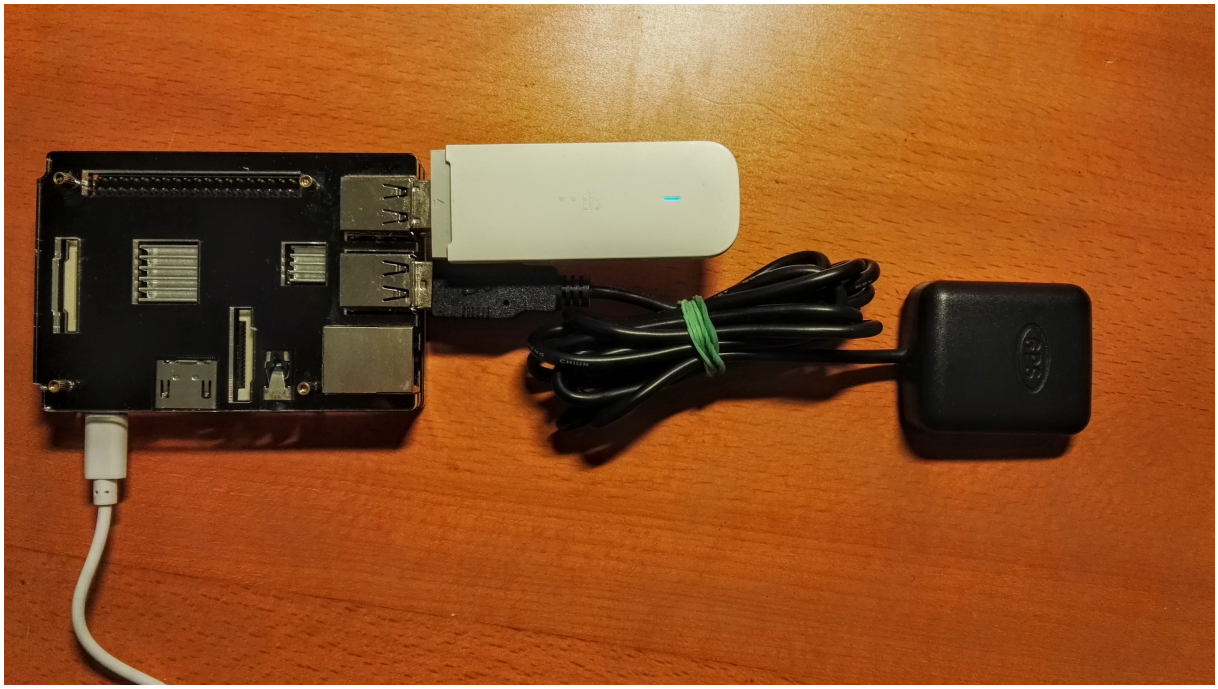
Celý systém by měl být dimenzován pro podporu více klientů, schopných operovat zcela nezávisle.

4.2 Realizace klientské části

Klientská část je postavena na jednodeskovém počítači Raspberry Pi 2 model B s operačním systémem Raspbian, a to především kvůli široké podpoře periferií, skriptovacího jazyka Python, a pořizovací ceně. Zprvu se jevil velmi dobře také systém OpenWRT, nicméně byl nahrazen právě systémem Raspbian, a to hlavně kvůli nemožnosti použít Python a podpoře proprietárního shellu Ash namísto shellu Bash. Fotku klientské části můžeme vidět na obrázku 6.

Tento počítač je dále vybaven LTE modemem Huawei E3372h-153 se SIM kartou operátora Vodafone CZ, GPS přijímačem Ublox UBX-G7020 a USB WiFi kartou TP-LINK TL-WN725N Ver:2.0.

- **Parametry LTE modemu**
  - LTE FDD: Band1(2100MHz)/Band3(1800MHz)/Band7(2600MHz)/ Band8(900MHz)/Band20(800MHz)
  - DC-HSPA+/HSPA+/HSPA/UMTS: Band1(2100MHz)/Band8(900MHz)
  - EDGE/GPRS/GSM: 850MHz/900MHz/1800MHz/1900MHz
- **Parametry GPS přijímače**
  - GPS/QZSS
  - GLONASS
- **Parametry WiFi karty**
  - IEEE 802.11bgn
  - Frekvenční pásmo 2.4 GHz



Obrázek 6: Fotka klientské části

4.2.1 Konfigurace operačního systému

Systém Raspbian byl ve své základní podobě dovybaven balíky pro konektivitu a práci s daty. Seznam všech balíků je uveden v příloze A. V případě periferií bylo nezbytné zajistit jejich okamžité fungování při zapojení či startu systému. LTE modem je nutné nejprve připojit do mobilní sítě a následně zaregistrovat, jak je uvedeno ve výpisu 1 z adresáře `/etc/udev/rules.d/70-usb-modeswitch.rules`.

```
ACTION == "add", SUBSYSTEM == "usb", ATTRS{idVendor}=="12d1", ATTRS{idProduct
    }=="14fe", RUN+="/usr/sbin/usb_modeswitch -v 12d1 -p 14fe -M
    '5553424312345678000000000000011062000000100000000000000000000'"
KERNEL=="wwan*", ATTRS{address}=="*", RUN+="/bin/sh /home/pi/connect.sh"
```

Výpis 1: Výpis z adresáře /etc/udev/rules.d/70-usb-modeswitch.rules

Druhý z uvedených příkazů spouští skript `/home/pi/connect.sh`, který zaregistruje modem do datové sítě, a následně požádá o přidělení IP adresy, jak znázorňuje výpis 2. Časová zpoždění zde mají pouze ochranný charakter.

```
#!/bin/bash -x
sleep 5
/bin/echo -e "AT^NDISDUP=1,1,\"internet\"\r" > /dev/ttyUSB0
sleep 1
/sbin/dhclient -v wwan0
```

```
exit 0
```

Výpis 2: Výpis z adresáře /home/pi/connect.sh

Po tomto kroku se modem stává datově aktivním a je připraven na sběr dat řídícím skriptem `/home/pi/modem-client.py` popsaným v sekci 4.2.4.

Spouštění řídícího skriptu po startu systému se v Debianových distribucích Linuxu, ke kterým patří i Raspbian, provádí v souboru `/etc/rc.local`. V tomto případě však `rc.local` spouští pouze pomocný skript, který se poté stará o spuštění hlavního programu. Změny provedené v `rc.local` jsou obsahem výpisu 3.

```
/home/pi/starter.sh || exit 1
exit 0
```

Výpis 3: Doplňkový kód v adresáři /etc/rc.local

Samotný `/home/pi/starter.sh` pak spouští řídící skript pro sběr dat, jak je uvedeno ve výpisu 4.

```
#!/bin/bash
python /home/pi/modem-client.py | /usr/bin/ts '[%Y-%m-%d %H:%M:%S]' &> /var/log
    /modem.log &
exit 0
```

Výpis 4: Výpis z adresáře /home/pi/starter.sh

Veškerý výstup ze skriptu `modem-client.py` je doplněn o časovou značku a uložen do rotujícího logu `/var/log/modem.log`.

4.2.2 Konfigurace WiFi AP

Jak již bylo zmíněno v úvodu této sekce, klientská část je vybavena WiFi rozhraním primárně určeným pro lokální bezdrátový přístup. Zvolená WiFi karta TP-LINK TL-WN725N Ver:2.0 je plně kompatibilní se systémem Raspbian pro základní síťové operace. Neplatí to však v případě Ad-Hoc módu, a proto bylo nutné doinstalovat potřebné balíky podle návodu na webových stránkách [9].

Po instalaci následovala pouze konfigurace síťového rozhraní `wlan0`, výpis 5.

```
auto wlan0
allow-hotplug wlan0
iface wlan0 inet static
  address 192.168.1.1
  netmask 255.255.255.0
  wireless-channel 2
  wireless-essid RaspPi-AP1
```

```
wireless-mode ad-hoc
```

Výpis 5: Nastavení WiFi AP v adresáři /etc/network/interfaces

Po připojení k nakonfigurované WiFi síti “RaspPi-AP1” je klient dostupný na IP adrese `192.168.1.1` a je povoleno také SSH.

RasperryPi ovšem nepodporuje žádné zabezpečení Ad-Hoc sítě, čímž se vytváří potenciální bezpečnostní díra v systému. Navíc je naistalovaný ovladač kompatibilní pouze s jednou verzí Linux kernelu a v případě “upgradu” se musí nainstalovat znovu podle dáného návodu. Z tohoto důvodu toto řešení není doporučeno na jiné, než testovací verzi.

4.2.3 Konfigurace OpenVPN

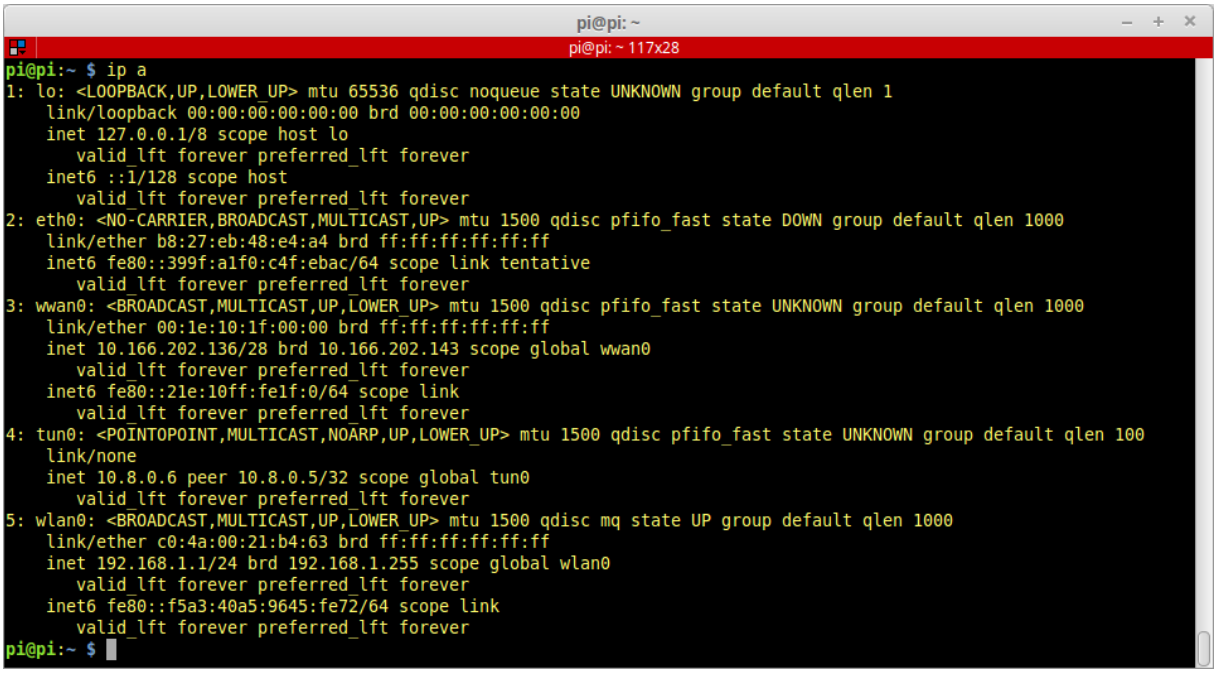
Jelikož při přenosu dat z klienta na server neprobíhá žádná autentizace, veškerý přenos je zabezpečen tunelovým VPN spojením. Klientskou konfiguraci VPN můžeme vidět ve výpisu 6.

```
client
dev tun
proto udp
remote 193.86.184.131 8194
resolv-retry infinite
nobind
persist-key
persist-tun
ca /etc/openvpn/ca.crt
cert /etc/openvpn/client1.crt
key /etc/openvpn/client1.key
ns-cert-type server
cipher AES-128-CBC
auth SHA1
tls-client
comp-lzo
verb 6
```

Výpis 6: Výpis z adresáře /etc/openvpn/client.conf

Druhým velkým důvodem je fakt, že O&M připojení je uskutečněno přes síť mobilního operátora, kde jsou veškerá IP zařízení "skryta"za NAT, a mají pouze privátní IP adresy. S použitím VPN se vytvoří tunel skrz NAT a klient je dostupný přímo ze serveru, ať už je jeho privátní IP adresa jakákoliv.

Kompletní nastavení síťových rozhraní na klientské části ukazuje obrázek 7.



Obrázek 7: Výpis příkazu `ip a`

4.2.4 Zpracování dat

Hlavní program klientské části tvoří Python skript `/home/pi/modem-client.py` spolu se svým konfiguračním souborem `/home/pi/register.conf`. Program je spouštěn pomocným skriptem `/home/pi/starter.sh`, jak je uvedeno v sekci 4.2.1.

Celý program je rozdělen do čtyř vláken:

- GPS
- MODEM
- SENDER
- CONNECT

Metoda `__main__` pak obsahuje pouze syntaxi pro inicializaci, spuštění vláken a cyklus `while`, který funguje jako prostředník mezi vlákny. Úkolem je vytvořit finální datový log ve stanovené jednotce času a uložit jej do slovníku, který slouží jako neseřazená fronta pro pozdější odeslání dat na server. Metoda `__main__` nejprve spustí všechna vlákna, a poté se dostane do hlavního cyklu, kde každé 2 sekundy vytvoří pole `data` z aktuálních dat a to poté uloží do slovníku `base`.

Kompletní zdrojový kód řídícího skriptu `/home/pi/modem-client.py` je obsahem přílohy C.

4.2.4.1 Vlákno GPS

Vlákno GPS je jako jediné inicializováno z konstruktoru své vlastní třídy `GpsPoller()`, která je volána z metody `__main__`. Jeho úkolem je vytvořit interní spojení s GPS deamonem[1] pro čtení parametrů z GPS, výpis 7. Ty jsou propagovány do výsledného logu skrz instanci `gpsd` v poli `data`.

```
class GpsPoller(threading.Thread):
   def __init__(self):
      threading.Thread.__init__(self)
      global gpsd #bring it in scope
      gpsd = gps(mode=WATCH_ENABLE) #starting the stream of info
      self.current_value = None
      self.running = True #setting the thread running to true
      self.modem_running = True #setting the thread running to true

   def run(self):
      global gpsd
      while gpsp.running:
         gpsd.next() #this will continue to loop and grab EACH set of gpsd info
               #to clear the buffer
```

Výpis 7: Třída GpsPoller skriptu modem-client.py

Zdrojový kód pro implementaci GPS v Pythonu by získán ze serveru GitHub [10].
Data z GPS jsou:

- Longitude - Zeměpisná délka [°]
- Latitude - Zeměpisná šířka [°]
- Altitude - Nadmořská výška [m]
- Speed - Pozemská rychlost [m/s]
- EPX - Chyba zeměpisné délky[2] [m]
- EPY - Chyba zeměpisné šířky[2] [m]
- EPV - Chyba nadmořské výšky[2] [m]
- EPS - Chyba pozemské rychlosti[2] [m/s]

4.2.4.2 Vlákno MODEM

Získávání parametrů z mobilní sítě obstarává vlákno MODEM, které se spouští příkazem `modemp.start()` v metodě `__main__`. To volá metodu `modemData()`, jejíž část můžeme vidět ve výpisu 8.

[1] Podpůrný program připojený přímo na sériovou linku GPS, který zpracovává NMEA data do strojově čitelného formátu pro řadu programovacích a skriptovacích jazyků. Deamon je součástí balíku `gpsd` v příloze A

[2] S přesností 95%

```
def modemData():
   global modem, register, network, csq, plmn, lte_hcsq, umts_hcsq, gsm_hcsq,
       creg, lte_hfreqinfo, gpsp, sending_enabled
   syscfg = None
   try:
      modem = serial.Serial(port='/dev/ttyUSB1', baudrate=115200, bytesize=8,
          parity='N', stopbits=1, timeout=0.6)
      while gpsp.modem_running:
         registerNetwork()
         try:
            modem.write("AT^HCSQ?\r")
            response = modem.readlines(40)
            #print response
            if len([k for k in response if "LTE" in k]) > 0:
               network = "LTE"
               print network
               lte_hcsq_tmp = parseValues(response)
               if lte_hcsq_tmp != None and len(lte_hcsq_tmp) == 5:
                  lte_hcsq_tmp = list(lte_hcsq_tmp)
                  #print lte_hcsq_tmp
                  lte_hcsq_tmp[1] = str(int(lte_hcsq_tmp[1]) - 120)
                  lte_hcsq_tmp[2] = str(int(lte_hcsq_tmp[2]) - 140)
                  lte_hcsq_tmp[3] = str(round(0.2 * int(lte_hcsq_tmp[3]) - 20,1)
                      )
                  lte_hcsq_tmp[4] = str(round(0.5 * int(lte_hcsq_tmp[4]) -
                      19.5,1))
                  lte_hcsq = lte_hcsq_tmp
               else:
                  lte_hcsq = ['LTE','','','','']
               #print "lte_hcqs:", lte_hcsq
               modem.write("AT^HFREQINFO?\r")
               hfreqinfo_tmp = parseValues(modem.readlines(64))
               if hfreqinfo_tmp != None and len(hfreqinfo_tmp) == 9:
                  lte_hfreqinfo = list(hfreqinfo_tmp)
               else:
                  lte_hfreqinfo = ['','','','','','','','','']
               #print "lte_hfreqinfo:", lte_hfreqinfo
            elif len([k for k in response if "WCDMA" in k]) > 0:
               network = "WCDMA"
               print network
               hcsq_tmp = parseValues(response)
               if hcsq_tmp != None and len(hcsq_tmp) == 5:
```

Výpis 8: Vlákno MODEM skriptu modem-client.py

V metodě se neprve spustí inicializace modemu ze sériového portu `/dev/ttyUSB1`, a poté se vytvoří smyčka `while`, která běží, dokud nedojde k přerušení programu příkazem `Ctrl+c` nebo ukončení vlákna GPS. Vše je ošetřeno výjímkou pro případ přerušení spojení se sériovým rozhraním. Následně se volá metoda `registerNetwork()`, jejímž úkolem je zkontrolovat obsah konfiguračního souboru `/home/pi/register.conf` a na jeho základě zaregistrovat modem do zvolené sítě1. Konfigurační soubor je kontrolován pro tyto hodnoty:

- 4G (Modem je zaregistrován pouze do sítě LTE)
- 3G (Modem je zaregistrován pouze do sítě UMTS/HSPA)

- 2G (Modem je zaregistrován pouze do sítě GSM)
- AUTO/"jiné"(Modem je automaticky zaregistrován do dostupné sítě nejvyšší kategorie)

Poznámka 1 Jelikož je obsah konfiguračního souboru kontrolován vždy na začátku cyklu, je možné měnit registraci za chodu programu,například příkazem:

```
echo 4G > /home/pi/register.conf
```

Hlavní problematikou tohoto vlákna bylo zajistit přiřazení příchozích dat ze sériové linky správné síti. V případě režimu "AUTO"se totiž může stát, že v čase mezi požadavkem a odpovědí modem změní síť. Příchozí hodnoty by poté neodpovídaly skutečnostem v dané sítí. Toto chování je ošetřeno vysláním požadavku `modem.write("AT^HCSQ? \r")` na začítku hlavního cyklu. Na základě příchozí odpovědi se pomocí podmínek `if` zvolí sekce v kódu pro danou síť. Výhodou tohoto řešení je, že lze měnit odesílané pořadavky v závislosti na typu sítě. Odpověď se následně zpracuje metodou `parseValues()`, která pomocí regulárních výrazů získává hodnoty ze sériové linky v požadovaném formátu.

Seznam všech použitých AT příkazů je obsahem přílohy D. Nutno podotknout, že v případě odpovědi na AT příkaz `AT^HCSQ?` metoda `parseValues()` nevrací reálné hodnoty parametrů sítě, ale pouze jejich reference. Hodnoty v dB nebo dBm proto musí být dopočítány. Příklad odpovědí a výpočtů můžeme vidět zde:

- **Formát 4G**
  - ^HCSQ:"LTE",**53**,45,181,18 — RSSI = "hodnota1"- 120 [dBm]
  - ^HCSQ:"LTE",53,**45**,181,18 — RSRP = "hodnota2"- 140 [dBm]
  - ^HCSQ:"LTE",53,45,**181**,18 — SINR = 0.2 x "hodnota3"- 20 [dB]
  - ^HCSQ:"LTE",53,45,181,**18** — RSRQ = 0.5 x "hodnota4"- 19.5 [dB]
- **Formát 3G**
  - ^HCSQ:"WCDMA",**53**,49,57 — RSSI = "hodnota1"- 120 [dBm]
  - ^HCSQ:"WCDMA",53,**49**,57 — RSCP = "hodnota2"- 120 [dBm]
  - ^HCSQ:"WCDMA",53,49,**57** — Ec/No = 0.5 x "hodnota3"- 32 [dB]
- **Formát 2G**
  - ^HCSQ:"GSM",**53** — RSSI = "hodnota1"- 120 [dBm]

Jakmile jsou požadované hodnoty dopočítány, uloží se do globálních proměnných, které jsou poté dostupné z metody `__main__`. Tím se pro tvorbu logů zajistí vždy nejaktuálnější informace. Na konci hlavního cyklu metody `modemData()` se nastaví globální proměnná `sending_enabled = 1`, která povolí odesílání dat na server.

4.2.4.3 Vlákno SENDER

Vlákno SENDER bylo sestaveno pro jediný úkol, a to dopravit data ze slovníku na server, a poté je z tohoto slovníku odstranit. Obsahem vlákna je opět metoda, ve které běží cyklus `while`.

V případě metody `sendData()` se jedná o cykly dva. Tím prvním je cyklus čekající na nastavení globální proměnné `sending_enabled`, jak je popsáno v sekci 4.2.4.2. Jakmile tato proměnná nabyde hodnoty 1, spustí se druhý cyklus, který získá hodnotu prvního klíče ze slovníku, načte data a odešle přes HTTP spojení na server. Pokud je kód HTTP odpovědi roven hodnotě 200, záznam daného klíče se vymaže ze slovníku.

Jak je vidět z výpisu kódu, příloha C, proces nekontroluje, jestli je databáze prázdná, ale používá syntaxi vyjímky `try/except`. Tento trik zajistí programu vyšší rychlost a zároveň sníží výpočetní náročnost.

Pokud je slovník prázdný, při čtení klíče nastane vyjímka `IndexError` a proces se uspí na 20 sekund. Pokud nastane jiný problém, např. nedostupnost serveru, vyjímky `BasStatusLine` a `URLError` jsou ošetřeny a program se uspí na 10 sekund. Oba tyto časové intervaly jsou velmi důležité a chrání celý cyklus před zahlcením a vyčerpáním veškerého dostupného výpočetního výkonu.

URL serveru, na kterou jsou data odesílána, je nastavena v proměnné `url` na začátku skriptu, příloha C.

4.2.4.4 Vlákno CONNECT

Poslední samostatný proces řídícího skriptu je vlákno CONNECT, které se v průběhu svého "života"stará o udržení internetového připojení. Při přeregistraci modemu totiž často dochází ke ztrátě konektivity a odesílání dat by už v tomto případě nebylo možné.

```
def connectToInternet():
   global gpsp
   time.sleep(10)
   while gpsp.running:
      if internetConnectionCheck() == False:
         print "Internet connection is down!"
         subprocess.call(["sudo", "killall", "dhclient"])
         subprocess.call(["sudo", "sh", "/home/pi/connect.sh"])
      else:
         print "Internet connection is up!"
      time.sleep(20)

def internetConnectionCheck():
   try:
      urllib2.urlopen('http://idnes.cz', timeout=1) #idnes.cz
      return True
   except urllib2.URLError as err:
      return False
```

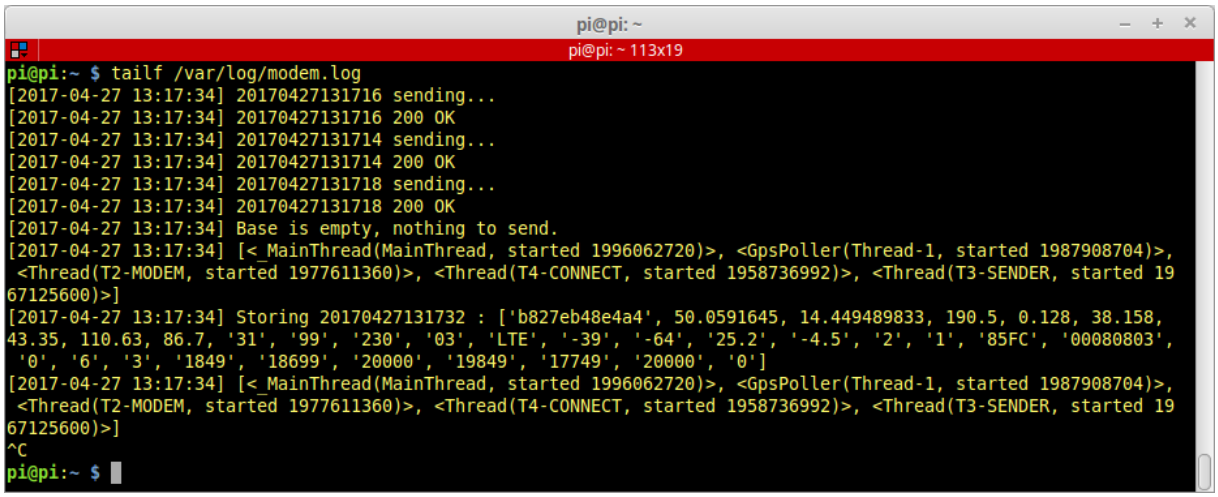
Výpis 9: Vlákno CONNECT skriptu modem-client.py

Vlákno znázorněné výpisem 9 spouští metodu `connectToInternet()`, která v intervalech 20 sekund kontroluje pomocí druhé metody `internetConnectionCheck()` dostupnost dané webové

stránky. Pokud je stránka nedospupná, ukončí se veškeré běžící `dhclient` procesy a zavolá se skript `/home/pi/connect.sh`, jehož funkce je popsána v sekci 4.2.1.

4.2.5 Logování standardního výstupu

Za účelem hledání a odstraňování problémů na klientské části byl implementován jednoduchý systém logování standardního výstupu `stdout` a `stderr` do souboru `/var/log/modem.log`. Logování je spouštěno se startem skriptu `modem-client.py`, jak ukazuje výpis 4. Strukturu logu můžeme vidět na obrázku 8.



Obrázek 8: Výpis z logu klientské části

Pro efektivní správu logů se využívá funkce `logrotate`, která je nastavena pro denní rotování tří souborů. Její konfigurace je znázorněna ve výpisu 10.

```
/var/log/modem.log {
    missingok
    daily
    create 0644 pi pi
    rotate 3
    copytruncate
}
```

Výpis 10: Výpis z adresáře /etc/logrotate.conf

4.3 Realizace serverové části

Serverová část je tvořena dvoujádrovým procesorem, 1 GB operační pamětí a kapacitou uložiště 40 GB. Celý server běží ve virtuálním prostředí. Operační systém je distrubuce Ubuntu Server 16.04 LTS a je skrytý za veřejnou IP adresou 193.86.184.131.

Jeho primárním účelem je přijímat, potvrzovat a ukládat data z klientů do databáze, a následně je prezentovat ve webovém rozhraní. Přenos dat je zabezpečen OpenVPN, jejíž serverová konfigurace je popsána v sekci 4.3.2. Webové rozhraní využívá Google Maps API a je napsáno ve skriptovacím jazyce PHP.

4.3.1 Konfigurace operačního systému

Server nevyžadoval mnoho zásahů do systému. Bylo však nezbytné zajistit konektivitu, spuštění potřebných aplikací při startu systému a také doinstalování balíků spojených s databází a webovým rozhraním.

4.3.1.1 Nastavení firewallu

Pro potřeby veškeré konektivity bylo nutné upravit nastavení firewallu, výpis 11.

```
martin@tali:~$ sudo ufw status
Status: active

To                         Action     From
--                         ------     ----
1194/tcp                   ALLOW      Anywhere
OpenSSH                    ALLOW      Anywhere
8005/tcp                   ALLOW      Anywhere
80/tcp                     ALLOW      Anywhere
1194/udp                   ALLOW      Anywhere
1194/tcp (v6)              ALLOW      Anywhere (v6)
OpenSSH (v6)               ALLOW      Anywhere (v6)
8005/tcp (v6)              ALLOW      Anywhere (v6)
80/tcp (v6)                ALLOW      Anywhere (v6)
1194/udp (v6)              ALLOW      Anywhere (v6
```

Výpis 11: Konfigurace Firewallu na serveru

Změny se týkaly portů TCP/UDP 1194 pro OpenVPN, TCP 80 pro webový server a TCP 8005 pro HTTP server zajšťující sběr dat od klientů. Veškeré tyto porty byly povoleny jak pro IPv4, tak pro IPv6.

4.3.1.2 Instalace balíků

Serverová část vyžadovala doinstalování balíků pro potřeby webového serveru Apache, databáze SQLite, PHP a OpenVPN, především tedy:

- **Webový server Apache**
  - apache2, apache2-bin, apache2-data, apache2-utils, libapache2-mod-php7.0
- **PHP**
  - php7.0, php7.0-cli, php7.0-common, php7.0-sqlite3, php7.0-xml
- **Databáze SQLite**

  - sqlite3, libsqlite0, libsqlite3-0

- **OpenVPN**
  - openvpn

Seznam všech instalovaných balíků je uveden v příloze B.

4.3.1.3 Automatické spouštění procesů

Stejně jako v případě klientské části, i na serveru je nezbytné po startu systému spustit veškeré potřebné procesy. Jelikož Ubuntu Server vychází z distribuce Debian, stejně jako v případě Raspbianu se spouštění aplikací provádí ze souboru `/etc/rc.local`. Zde se konfigurace v zásadě neliší a opět je použit pomocný skript `starter.sh`, jak je znázorněno ve výpisu 12.

```
/home/martin/starter.sh || exit 1
exit 0
```

Výpis 12: Výpis ze souboru /etc/rc.local

Skript `starter.sh` v domovském adresáři uživatele `martin` pak spouští na pozadí HTTP server `modem-server.py` a agregační skript `modem-aggregator.py`, jak ukazuje výpis 13.

```
#!/bin/bash
python /home/martin/modem-server.py --debug --verbose &>/var/log/modem-server.
    log &
python /home/martin/modem-aggregator.py &
exit 0
```

Výpis 13: Výpis ze souboru /home/martin/starter.sh

4.3.2 Konfigurace VPN

VPN má v řešení této práce velmi zásadní úkol. Nejen že zabezpečuje veškerý přenos dat, ale poskytuje také přímé spojení s klientem. Jak už bylo zmíněno v sekci 4.2.3, klient je pro přístup ze sítě schován za NAT mobilního operátora. S pomocí VPN se nám proto otevírá cesta, jak spravovat konfiguraci klienta vzdáleně.

Nastavení VPN serveru je znázorněno ve výpisu 14.

```
port 1194
proto udp
dev tun
ca /etc/openvpn/ca.crt
cert /etc/openvpn/server.crt
key /etc/openvpn/server.key
dh /etc/openvpn/dh2048.pem
server 10.8.0.0 255.255.255.0
ifconfig-pool-persist ipp.txt
keepalive 10 120
cipher AES-128-CBC # AES
auth SHA1
comp-lzo
user nobody
```

```
group nogroup
persist-key
persist-tun
status /tmp/openvpn-status.log
log         openvpn.log
log-append  openvpn.log
verb 5
```

Výpis 14: Konfigurace OpenVPN ze souboru /etc/openvpn/server.conf

Jak je zřejmé, je snahou udržet spojení vždy aktivní parametrem `persist-tun` a přiřazovat klientům stále stejné IP adresy z dostupného poolu. Spojení je zabezpečeno 128 bitovým AES šifrováním a autentizace paketů využívá hashovací funkci SHA1. Veškerý přenos je dále komprimován parametrem `comp-lzo`.

4.3.3 HTTP server pro sběr dat

Jedním z hlavních prvků serverové části je Python skript `/home/martin/modem-server.py`. Ten pracuje jako HTTP server poslouchající na lokálním portu 8005. Jakmile přijde HTTP požadavek na tento port, program z něj získá potřebná data, uloží je do tabulky Data databáze SQLite v adresáří `/var/www/html/modem-database.sqlite`, a následně odešle odpověď HTTP 200 OK zpátky na odesílatele. V případě, že se data nepodaří uložit, použije se kód HTTP 400 Bad Request.

Skript je složen ze dvou tříd a dalších pomocných metod a je spouštěn z metody `__main__`, výpis 15. Ta se mimo definici a kontrolu parametrů při spuštění pouze připojí k databázi a vytvoří instance třídy `Pi2RequestHandler` a třídy `dataObject`, které se dále starají o veškerou činnost.

```
    global LC
    debug = False
    verbose = False
    logFile = ''
    db = '/var/www/html/modem-database.sqlite'

    listenPort = 8005
    listenAddress = '0.0.0.0'
    try:
        opts, args = getopt.getopt(sys.argv[1:],
                                   "",
                                   ["help",\
                                    "verbose",\
                                    "debug",\
                                    "db="\
                                    "log-file=",\
                                    "port=",\
                                    "address="])

    except getopt.GetoptError, err:
        print str(err) # will print something like "option -a not recognized"
        usage()
        sys.exit(2)
```

```
for o, a in opts:
    if o in ("--debug"):
        debug = True
    elif o in ("--verbose"):
        verbose = True
    elif o in ("--db"):
        db = a
    elif o in ("--log-file"):
        logFile = a
    elif o in ("--port"):
        listenPort = a
    elif o in ("--address"):
        listenAddress = a
    else:
        assert False, "unhandled option"

if verbose:
    log.setLevel(logging.INFO)

if debug:
    log.setLevel(logging.DEBUG)

if len(logFile) != 0 :
    fh = logging.FileHandler(logFile)
    fh.setLevel(logging.DEBUG)
    fh_formatter = logging.Formatter("%(asctime)s - %(name)s - %(levelname)
        s - %(message)s")
    fh.setFormatter(fh_formatter)

log.info('Initializing monkey with SQLite(%s)' %(lite.sqlite_version))

try:
    log.debug('Connecting db: %s' %(db))
    LC = lite.connect(db)
except lite.Error, e:
    log.error('SQLite exception caught %s' %(e))
    sys.exit(1)

log.debug('Create Database structure')
dataObject.createSQLTable(LC)

log.debug('Starting server %s:%d' %(listenAddress,listenPort))

server = BaseHTTPServer.HTTPServer((listenAddress,listenPort),
                                   Pi2RequestHandler)

try:
        server.serve_forever()
except KeyboardInterrupt as e:
        sys.stdout.write('Keyboard interrupt\n')
finally:
        server.shutdown()
        server.server_close()
        LC.close()
```

Výpis 15: Metoda main skriptu modem-server.py

Třída `Pi2RequestHandler` má definovány tři metody. V našem případě je však použita pouze metoda `do_POST()`, výpis 16, která v případě přijetí HTTP POST požadavku data převede do

objektů třídy `dataObject` a dále vloží do tabulky a odešle HTTP odpověď.

```
class Pi2RequestHandler(BaseHTTPServer.BaseHTTPRequestHandler):
        def do_HEAD(s):
                log.debug('Got HEAD request')
                s.send_response(200)
                s.send_header("Content-type", "text/html")
                s.end_headers()
        def do_GET(s):
                """Respond to a GET request."""
                log.debug('Got GET request')
                s.send_response(400)
                s.send_header("Content-type", "text/plain")
                s.end_headers()
                s.wfile.write("Invalid request")
        def do_POST(s):
                global LC
                """Respond to a POST request."""
                log.debug('Got POST request')
                length = int(s.headers['Content-Length'])

                dataObj = dataObject.fromHttpPost(urlparse.parse_qs(s.rfile.read
                    (length).\
                         decode('utf-8')))

                log.debug(dataObj)

                if dataObj is not None:
                    dataObj.insertIntoSQL(LC)
                    LC.commit()
                    s.send_response(200)
                else:
                    s.send_response(400)

                s.send_header("Content-type", "text/plain")
                s.end_headers()
```

Výpis 16: Třída Pi2RequestHandler skriptu modem-server.py

Třída `dataObject` už pouze obsahuje již volané metody, a to `createSQLTable()`, která zajistí vytvoření tabulky v databázi, metodu `insertIntoSQL()`, která z přijatého objektu odstraní nežádoucí znaky, převede na textový formát a vloží do tabulky Data a metodu `fromHttpPost()`, která zajišťuje parsování dat z HTTP POST požadavku do objektu své třídy, výpis 17.

```
class dataObject:
    recId = -1
    transactionId = unicode('<unassigned>')
    #timestamp = unicode('<unassigned>')
    values = unicode('<unassigned>')
    @staticmethod
    def createSQLTable(lc):
        cur = lc.cursor()
        try:
            log.info("Create SQL Database Table Data")
            #cur.execute("DROP TABLE IF EXISTS Data")
            cur.execute("CREATE TABLE IF NOT EXISTS Data(timestamp TEXT NOT NULL
                ,"\
                    "device TEXT NOT NULL,lat TEXT NOT NULL,lon TEXT NOT NULL
                        ,alt TEXT,"\
```

```
                    "speed TEXT,epx TEXT,epy TEXT,epv TEXT,eps TEXT,csq TEXT
                        NOT NULL,"\
                    "ber TEXT,plmn_cc TEXT,plmn_nc TEXT,network TEXT NOT NULL
                        ,"\
                    "rssi TEXT NOT NULL,rssp_rscp TEXT,sinr_ecno TEXT,rsrq
                        TEXT,reg_mode TEXT,"\
                    "reg_stat TEXT,lac TEXT,cellid TEXT,zero TEXT,six TEXT,
                        lte_band TEXT,"\
                    "earfcn_dl TEXT,freq_dl TEXT,bw_dl TEXT,earfcn_ul TEXT,
                        freq_ul TEXT,"\
                    "bw_ul TEXT,aggr_num TEXT)")
        except lite.Error, e:
            log.error('Creating SQLite exception caught %s' %(e))
            sys.exit(1)

    def insertIntoSQL(s,lc):
        log.debug('Inserting transaction-id [%s] into the DB' %(s.transactionId
            ))
        cur = lc.cursor()
        try:
            cur.execute("INSERT OR REPLACE INTO Data VALUES
                (?,?,?,?,?,?,?,?,?,?,?,"\
                        "?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)",\
                        str(s.all).translate(None, "[] '").split(','))

           # s.recId = cur.lastrowid
        except lite.Error, e:
            log.error('SQLite exception caught %s' %(e))
            sys.exit(1)
        log.debug('inserted %d' %(s.recId))


    @staticmethod
    def fromHttpPost(p_data):
        log.debug(p_data)
        obj = None
        for key, value in p_data.iteritems():
            log.debug("%s=%s" % (key, value))
            obj = dataObject()
            #obj.transactionId = p_data['transaction-id'][0]
            obj.transactionId = p_data['transaction-id'][0]
            #obj.time         = p_data['timestamp'][0]
            obj.values       = str(p_data['data'][0])
            obj.all          = obj.transactionId + ',' + obj.values
        return obj
```

Výpis 17: Třída dataObject skriptu modem-server.py

Jako `obj.transactionId` je použita hodnota `timestamp`. Kompletní kód je dostupný v příloze E.

4.3.3.1 Datový formát

Data z klientů jsou ukládána do tabulky `Data` databáze SQLite přímo ve webovém adresáři `/var/www/html/modem-database.sqlite`. Tabulka se indexuje podle čísla řádku, tzv. `rowid`, které je unikátní pro každý záznam. Strukturu dat znázorňuje tabulka 1.

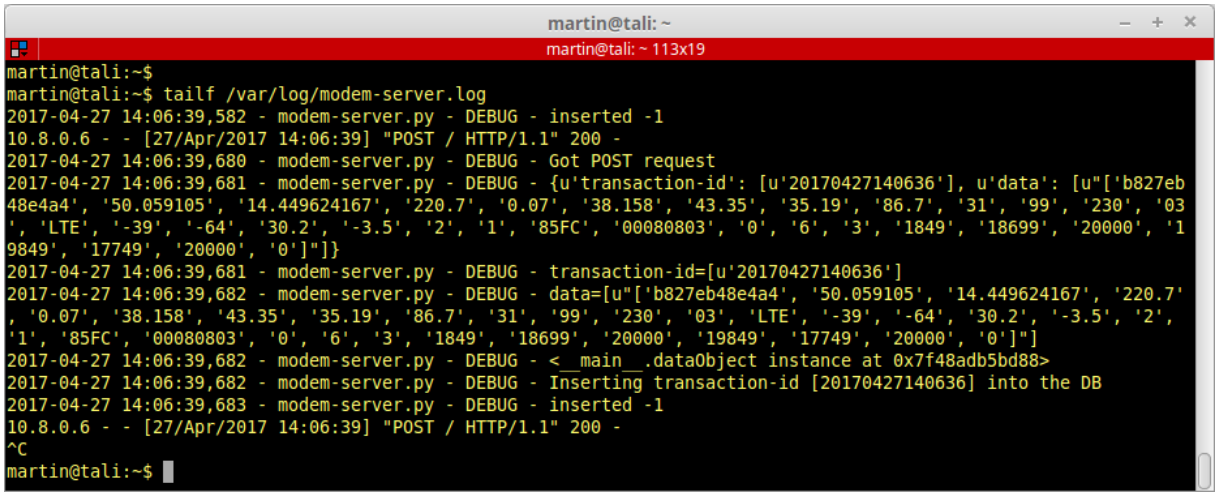
Pole `rowid` není součástí měřených dat a v tabulce se automaticky nachází na pozici "0". Zvláštní význam má pole `aggr_num`, jehož funkce je blíže popsána v sekci 4.3.4.

Tabulka 1: Formát databázové tabulky Data

Poř.	**Pole**	**Popis**	**Poř.**	**Pole**	**Popis**
1	timestamp	časová značka, formát YYYYMMDDHHMMSS	18	sinr_ecno	LTE: SINR [dB], WCDMA: Ec/No [dB]
2	device	MAC adresa eth0 klienta	19	rsrq	RSRQ [dB] [4]
3	lat	zeměpisná šířka [°]	20	reg_mode	0-Zakázáno, 1-Povoleno, 2-Povoleno s informacemi o poloze
4	lon	zeměpisná délka [°]	21	reg_stat	1-Domácí síť, 5-Roaming
5	alt	nadmořská výška [m]	22	lac	Location Area Code 4 char HEX
6	speed	pozemská rychlost [m/s]	23	cellid	CellID 4-8 char
7	epx	chyba zem. délky [m]	24	zero	"0" -bez známého významu
8	epy	chyba zem. šířky [m]	25	six	"6" -bez známého významu
9	epv	chyba nadm. výšky [m]	26	lte_band	LTE pásmo, např. "20" = Band20[4]
10	eps	chyba poz. rychlosti [m/s]	27	earfcn_dl	downlink EARFCN[4]
11	csq	kvalitativní parametr signálu[3]	28	freq_dl	downlink frekvence např. "8110" = 811.0 MHz[4]
12	ber	bit error rate	29	bw_dl	downlink šířka pásma např. "5000" = 5 MHz[4]
13	plmn_cc	country code	30	earfcn_ul	uplink EARFCN[4]
14	plmn_nc	network code	31	freq_ul	uplink frekvence např. "8520" = 852.0 MHz[4]
15	network	síť LTE/ WCDMA/ GSM	32	bw_ul	uplink šířka pásma např. "5000" = 5 MHz[4]
16	rssi	RSSI [dBm]	33	aggr_num	agregační číslo
17	rsrp_rscp	LTE: RSRP [dBm], WCDMA: RSCP [dBm]			

4.3.3.2 Logování standardního výstupu

Stejně jako v případě klientské části, i na serveru je implementována funkce pro logování standardních výstupů. Logy se nacházejí v adresáři `/var/log/modem-server.log*` a jejich výstup je znázorněn na obrázku 9.



Obrázek 9: Výpis z logu serverové části

Na serveru se také využívá aplikace `logrotate`, jejíž konfigurace je zobrazena ve výpisu 18.

```
/var/log/modem-server.log {
    missingok
    daily
    create 0664 martin syslog
    rotate 10
    copytruncate
}
```

Výpis 18: Výpis z adresáře /etc/logrotate.conf

4.3.4 Agregace dat

V průběhu vývoje webového rozhraní byly zjištěny značné limity při práci s velkým množstvím dat, řádově stovkami tisíc. Google Maps API nebylo schopné zobrazit více než deset tisíc bodů bez náznaků zpoždění a webová stránka začínala být velmi nepřehledná. Z tohoto důvodu se mezi surová data z klientů uložená v tabulce Data a webové rozhraní vložil agregační článek kumulující informace z více měřenych dat a připravující nová, již agregovaná data pro zobrazení na mapě.

[3] Bezrozměrná veličina odpovídající tabulkovým hodnotám úrovně RSSI
[4] Pouze pro LTE síť

Veškerou agregaci obstarává Python skript `/home/martin/modem-aggregator.py`. Ten se při své inicializaci spojí s databází, jejíž adresa je uložena v proměnné `db`, výpis 19. Následně se pokusí vytvořit tabulku Aggr, do které se dále budou ukládat agregovaná data, a spustí hlavní cyklus celého programu. Na jeho začátku se nejprve získá počet ještě neagregovaných dat z tabulky Data. Ty se vyznačují hodnotou "0" v poli `aggr_num`. V případě, že počet neagregovaných dat menší než jedna, program se uspí na 30 sekund a operace se poté opakuje.

V případě, že tabulka Data obsahuje data, která by se dala agregovat, obsah celé tabulky Aggr se načte do slovníku `aggr_base` (za předpokladu, že ve slovníku ještě není z předchozích cyklů), a prvních 10000 neagregovaných dat z tabulky Data se načte do slovníku `data_base`.

```
conn = sqlite3.connect(db)
cur = conn.cursor()
cur.execute("CREATE TABLE IF NOT EXISTS Aggr(timestamp TEXT NOT NULL,device
    TEXT NOT NULL,"\
            "lat TEXT NOT NULL,lon TEXT NOT NULL,alt TEXT,speed TEXT,epx TEXT,
                epy TEXT,"\
            "epv TEXT,eps TEXT,csq TEXT NOT NULL,ber TEXT,plmn_cc TEXT,plmn_nc
                TEXT,"\
            "network TEXT NOT NULL,rssi TEXT NOT NULL,rsrp_rscp TEXT,sinr_ecno
                TEXT,"\
            "rsrq TEXT,reg_mode TEXT,reg_stat TEXT,lac TEXT,cellid TEXT,zero
                TEXT,six TEXT,"\
            "lte_band TEXT,earfcn_dl TEXT,freq_dl TEXT,bw_dl TEXT,earfcn_ul TEXT
                ,freq_ul TEXT,"\
            "bw_ul TEXT,aggr_num TEXT)")

newloop = 0

while True:
  cur.execute("SELECT COUNT(*) FROM Data WHERE aggr_num=0")
  pending_aggregation = cur.fetchone()[0]
  if pending_aggregation < 1:
    print "Records pending aggregation:", pending_aggregation
    print "Sleeping...30s"
    time.sleep(30)
  else:
    print "Records pending aggregation:", pending_aggregation

    if bool(aggr_base) == False:
      for row in cur.execute("SELECT rowid,* FROM Aggr"):
        aggr_base[row[0]] = row[1:]

    for row in cur.execute("SELECT rowid,* FROM Data WHERE aggr_num like ?
        limit 10000", (0,)):
      data_base[row[0]] = row[1:]
```

Výpis 19: Inicializace skriptu modem-aggregator.py

Takto připravená data se postupně vyčítají dvěma vnořenými cylky `for` za účelem nalezení shody, jak je znázorněno ve výpisu 20. Ta spočívá ve shodě v polích `device`, `network` a `cellid` a zároveň v zeměpisné poloze, kde vzdálenost bodů musí být menší než 0.03 km. Výpočet vzdálenosti obstarává metoda `haversine()` použitá z webových stránek [11].

Pokud dojde ke zmíněné shodě, z původních dat se vytvoří data nová následujícím způsobem:

- **Pole ponechána z tabulky Aggr:**
  - `plmn_cc`, `plmn_nc`, `network`, `reg_mode`, `reg_stat`, `lac`, `cellid`
- **Pole vypočtena aritmetickým průměrem mezi tabulkou Data a Aggr:**
  - `lat`, `lon`, `alt`, `speed`, `epx`, `epy`, `epv`, `eps`, `csq`, `ber`, `rssi`, `rsrp_rscp`, `sinr_ecno`, `rsrq`
- **Pole nahrazena z tabulky Data:**
  - `timestamp`
- **Pole vyprázdněna:**
  - `lte_band`, `earfcn_dl`, `freq_dl`, `bw_dl`, `earfcn_ul`, `freq_ul`, `bw_ul`
- **Pole inkrementována:**
  - `aggr_num`

Tato agregovaná data následně nahradí původní záznam ve slovníku `aggr_base` odpovídající příslušnému klíči `rowid`. Navíc se vytvoří reference v poli `aggr_num` slovníku `data_base`, která odpovídá poli `rowid` v budoucím záznamu tabulky Aggr, jak je uvedeno v poznámce 2.

Poznámka 2 V tabulce Aggr odpovídá pole `aggr_num` počtu surových dat, ze kterých byl záznam vytvořen. V tabulce Data odpovídá stejné pole číslu `rowid` v tabulce Aggr, jako reference mezi oběma záznamy. Z toho plyne, že v tabulce Aggr nemůže nabývat pole `aggr_num` nižší hodnoty než "1". Hodnota "0" v tabulce Data naopak označuje záznam, který ještě nebyl agregován.

```
    for dkey in data_base.keys():     # For ALL in Data
      newloop = 0
      for akey in aggr_base.keys():   # For ALL in Aggr
        if (round(haversine(float(data_base[dkey][2]),float(data_base[dkey][3])
            ,\
            float(aggr_base[akey][2]),float(aggr_base[akey][3])), 2) < 0.03 and\
            data_base[dkey][1]==aggr_base[akey][1] and data_base[dkey][14]==
                aggr_base[akey][14] and\
            data_base[dkey][22]==aggr_base[akey][22]):
          if aggr_base[akey][32] == None or int(aggr_base[akey][32]) < 1:
            num = 1
          else:
            num = int(aggr_base[akey][32])+1
          #print "num", num
          newrow = [akey] + list(data_base[dkey][0:25]) + [None,None,None,None,
              None,None,None,num]
                                                    # Create new record for
                                                        Aggr
          to_be_mean = [3,4,5,6,7,8,9,10,11,12,16,17,18,19]
          for pos in to_be_mean:
```

```
          try:
            if (pos == 3 or pos == 4):
              newrow[pos] = round(mean([float(data_base[dkey][pos-1]),\
                            float(aggr_base[akey][pos-1])]),7)
            elif (pos == 5):
              newrow[pos] = round(mean([float(data_base[dkey][pos-1]),\
                            float(aggr_base[akey][pos-1])]),1)
            elif (pos == 6):
              newrow[pos] = round(mean([float(data_base[dkey][pos-1]),\
                            float(aggr_base[akey][pos-1])]),6)
            elif (pos in range(7,11)):
              newrow[pos] = round(mean([float(data_base[dkey][pos-1]),\
                            float(aggr_base[akey][pos-1])]),2)
            else:
              newrow[pos] = int(round(mean([float(data_base[dkey][pos-1]),\
                            float(aggr_base[akey][pos-1])]),0))
          except (TypeError, NameError, ValueError) as e:
            #print "Error:", e
            newrow[pos] = data_base[dkey][pos-1]

        # print "NR:", newrow
        # print "Ndata:", [dkey] + list(data_base[dkey][:-1]) + [akey]
         aggr_base[akey] = newrow[1:]        # Save new Aggr recors
         data_base[dkey] = list(data_base[dkey][:-1]) + [akey]
         newloop = 1
         break

      if newloop == 1:
        newloop = 0
        continue
```

Výpis 20: Porovnání dat ve skriptu modem-aggregator.py

Jak ukazuje výpis 21, za předpokladu, že záznam není shodný s žádným záznamem z tabulky Aggr, je tento vložen přímo do slovníku pod klíčem `rowid` o jedno výšším, než je nejvyšší hodnota klíče, a pole `aggr_num` je inkrementováno o jedna.

Jakmile jsou všechna načtená data zpracována, vytvoří se tabulka AggrTemp, do které se nahraje obsah slovníku `aggr_base` a ta následně nahradí tabulku Aggr, jejíž data slouží pro prezentaci na webové stránce. Originální záznamy v tabulce Data jsou následně modifikovány o změny v poli `aggr_num`.

```
      if bool(aggr_base.keys()) == False:
        newkey = 1
      else:
        newkey = max([int(x) for x in aggr_base.keys()])+1
      #print "aggr2", [newkey] + list(data_base[dkey][:-1]) + [1]
      #print "data2", [dkey] + list(data_base[dkey][:-1]) + [newkey]
      aggr_base[newkey] = list(data_base[dkey][:-1]) + [1]
      data_base[dkey] = list(data_base[dkey][:-1]) + [newkey]

    # WRITE DOWN ALL CHANGES
    try:
      cur.execute("DROP TABLE AggrTemp")
    except Exception as e:
      print e
      #quit()
```

```
cur.execute("CREATE TABLE IF NOT EXISTS AggrTemp(timestamp TEXT NOT NULL,
    device TEXT NOT NULL,"\
            "lat TEXT NOT NULL,lon TEXT NOT NULL,alt TEXT,speed TEXT,epx
               TEXT,epy TEXT,"\
            "epv TEXT,eps TEXT,csq TEXT NOT NULL,ber TEXT,plmn_cc TEXT,
               plmn_nc TEXT,"\
            "network TEXT NOT NULL,rssi TEXT NOT NULL,rsrp_rscp TEXT,
               sinr_ecno TEXT,"\
            "rsrq TEXT,reg_mode TEXT,reg_stat TEXT,lac TEXT,cellid TEXT,zero
                TEXT,six TEXT,"\
            "lte_band TEXT,earfcn_dl TEXT,freq_dl TEXT,bw_dl TEXT,earfcn_ul
               TEXT,freq_ul TEXT,"\
            "bw_ul TEXT,aggr_num TEXT)")

print "aggr1", aggr_base[1]
for akey in aggr_base.keys():
  newrow = [akey] + list(aggr_base[akey])
  cur.execute("INSERT OR REPLACE INTO AggrTemp(rowid,timestamp,device,lat,
      lon,alt,speed,"\
             "epx,epy,epv,eps,csq,ber,plmn_cc,plmn_nc,network,rssi,
                rsrp_rscp,"\
             "sinr_ecno,rsrq,reg_mode,reg_stat,lac,cellid,zero,six,lte_band
                ,"\
             "earfcn_dl,freq_dl,bw_dl,earfcn_ul,freq_ul,bw_ul,aggr_num)"\
             "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,"\
             "?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)", newrow)
try:
  cur.execute("DROP TABLE Aggr")
except Exception as e:
  print e
  #quit()
cur.execute("ALTER TABLE AggrTemp RENAME TO Aggr")
#print "NumOfKeysInDataBase:", len(data_base.keys())
for dkey in data_base.keys():
  #print 'dkey:', dkey
  cur.execute("UPDATE Data SET aggr_num=? WHERE rowid=?", [data_base[dkey
      ][32],dkey])
dbCommit()      # Save to disc
data_base = {}
```

Výpis 21: Zápis dat ve skriptu modem-aggregator.py

Kompletní zdrojový kód agregačního skriptu `/home/martin/modem-aggregator.py` je obsahem přílohy F.

4.3.5 Prezentace měřených dat

Prezentace měřených a již agregovaných dat probíhá na webové stránce běžící na HTTP serveru Apache. Stránka je napsána ve skriptovacím jazyku PHP a je dostupná na standardním portu 80. Pro zobrazení dat byla využita API Google Maps a web je optimalizován pro prohlížeč Google Chrome.

4.3.5.1 Zdrojový kód

Zdrojový kód webové stránky se nachází v souboru`/var/www/html/index.php` a ve své HTML

hlavičce využívá hostované knihovny jQuery Core, jQuery UI a styly dostupné z [12] [13]. Z důvodu optimalizace rychlosti mapy a celého webu byl tento skript rozšířen o funkci MarkerClusterer, která efektivně shlukuje body na mapě při jejím oddálení. Funkce pracuje s javascriptovou knihovnou `markerclusterer.min.js` umístěnou ve stejném webovém adresáři. Knihovna je minimalizovanou verzí Google Maps MarkerClusterer a byla získána ze serveru GitHub [14].

V první části kódu jsou definovány styly jednotlivých použitých komponent a také implementovány již zmiňované hostované knihovny. Skript se následně připojí k SQLite databázi přes rozhraní PDO a při prvním načtení stránky zobrazí posledních tisíc hodnot z tabulky Aggr, jak znázorňuje výpis 22.

```
if (!isset($_GET['showMap'])) {
    $query = "SELECT rowid,* FROM Aggr ORDER BY timestamp DESC LIMIT 1000";
    $id = 0;
    foreach ($dbh->query($query) as $row) {
        $id++;
        if ($id == 1) {
            $minTime = $row['timestamp'];
            $maxTime = $row['timestamp'];
        } else {
            if ($minTime > $row['timestamp']) $minTime = $row['timestamp'];
            if ($maxTime < $row['timestamp']) $maxTime = $row['timestamp'];
        }
    }
    $year = substr($minTime, 0, 4);
    $mounth = substr($minTime, 4, 2);
    $day = substr($minTime, 6, 2);
    $hours = substr($minTime, 8, 2);
    $minutes = substr($minTime, 10, 2);
    $seconds = substr($minTime, 12, 2);
    $_GET['from'] = $day . '.' . $mounth . '.' . $year . ' ' . $hours . ':'
        . $minutes . ':' . $seconds;

    $year = substr($maxTime, 0, 4);
    $mounth = substr($maxTime, 4, 2);
    $day = substr($maxTime, 6, 2);
    $hours = substr($maxTime, 8, 2);
    $minutes = substr($maxTime, 10, 2);
    $seconds = substr($maxTime, 12, 2);
    $_GET['to'] = $day . '.' . $mounth . '.' . $year . ' ' . $hours . ':' .
        $minutes . ':' . $seconds;

}
```

Výpis 22: Inicializace webové stránky

Jelikož je formát pole timestamp v tabulce Aggr primárně určen pro čtení ze strany uživatele, pro funkci stránky se musel přepočítat na Unixový čas[3]. Ten je použitý i pro funkci posuvníku pro manuálního nastavení času.

Implementace mapy Google Maps spolu s funkcí MarkerClusterer je zobrazena ve výpisu 23. Zde se vytvoří instance mapy a jako výchozí pozice se použije poslení známá poloha. Poté se

[3]Unixový čas, angl. *"Unix epoch time"*, je formát času v sekundách od koordinovaného světového času (UTC) 00:00:00 1. ledna 1970.

cyklem `for` vykreslí ostatní body na mapu. Metoda `addListener` sloužící pro zobrazení detailů o zvoleném bodu skryje toto menu a čeká na zvolení bodu, pro který se má menu zobrazit. Její další metody pak kontrolují změnu středu mapy a přiblížení a starají se o opětovné vykreslení bodů.

```
function initMap() {
    var uluru = {lat: Number(startLocation1), lng: Number(
        startLocation2)};
    map = new google.maps.Map(document.getElementById('map'), {
        zoom: 15,
        center: uluru
    });
    for (i = 0; i < locations.length; i++) {
        marker = new google.maps.Marker({
            position: new google.maps.LatLng(locations[i][1],
                locations[i][2]),
            map: map,
            title: locations[i][4],
            zIndex: Number(locations[i][0]),
            icon: {
                path: "M27.648 -41.399q0 -3.816 -2.7 -6.516t-6.516
                     -2.7 -6.516 2.7 -2.7 6.516 2.7 6.516 6.516
                    2.7 6.516 -2.7 2.7 -6.516zm9.216 0q0 3.924
                    -1.188 6.444l-13.104 27.864q-0.576 1.188 -1.71
                     1.872t-2.43 0.684 -2.43 -0.684 -1.674 -1.872l
                    -13.14 -27.864q-1.188 -2.52 -1.188 -6.444 0
                    -7.632 5.4 -13.032t13.032 -5.4 13.032 5.4 5.4
                    13.032z",
                scale: 0.5,
                strokeWeight: 0.2,
                strokeColor: 'black',
                strokeOpacity: 1,
                fillColor: locations[i][3],
                fillOpacity: 0.85
            }
        });
        markers[i] = marker;

        google.maps.event.addListener(marker, 'click', (function
            (marker, i) {
            return function () {
                $('.map-marker-block').hide();
                $('#map-marker-' + marker.getZIndex()).show();
            }
        })(marker, i));
    }
    map.addListener('center_changed', function () {
        $('.title').hide();
        for (var i = markers.length, bounds = map.getBounds(); i
            --;) {
            if (bounds.contains(markers[i].getPosition())) {
                $('.title-' + markers[i].getZIndex()).show();
            }
        }
    });
    map.addListener('zoom_changed', function () {
        $('.title').hide();
```

```
            for (var i = markers.length, bounds = map.getBounds(); i
                --;) {
                if (bounds.contains(markers[i].getPosition())) {
                    $('.title-' + markers[i].getZIndex()).show();
                }
            }
        });
        // Add a marker clusterer to manage the markers.
        markerCluster = new MarkerClusterer(map, markers,
            {imagePath: 'https://developers.google.com/maps/
                documentation/javascript/examples/markerclusterer/m',
                minimumClusterSize: 5, maxZoom: 13});
        google.maps.event.addListenerOnce(map, 'idle', function(){
            $('.title').hide();
            for (var i = markers.length, bounds = map.getBounds(); i
                --;) {
                if (bounds.contains(markers[i].getPosition())) {
                    $('.title-' + markers[i].getZIndex()).show();
                }
            }
        });
    }
```

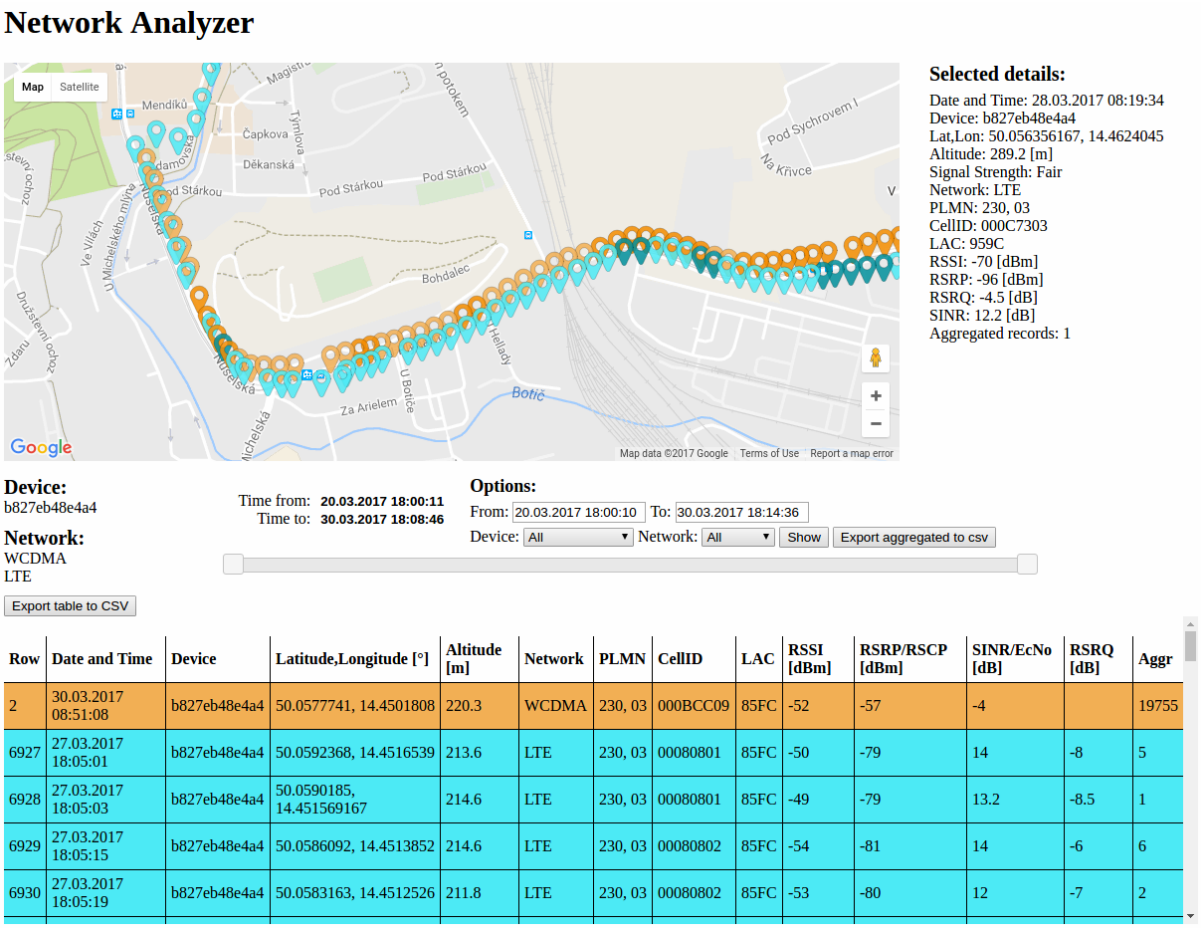
Výpis 23: Implementace Google Maps s funkcí MarkerClusterer

Instance `markerCluster` má na starosti již zmiňované shlukování bodů. Parametr `minimumClusterSize` udává minimální počet bodů, `maxZoom` pak počáteční velikost oddálení, pro které se body na mapě začnou shlukovat.

Zdrojový kód celé webové stránky je dostupný v příloze G.

4.3.5.2 Webová stránka

Webová stránka je ve formě jednostránkové prezentace. Je vytvořena jednoduše a přehledně s důrazem především na celkovou funkčnost. Stránka je tvořena mapou zobrazující body měření, ovládacími prvky a tabulkou obsahující údaje o jednotlivých bodech, obrázek 10.



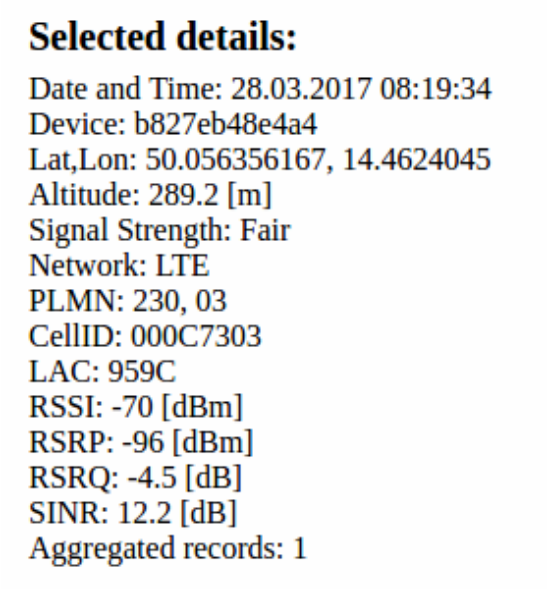
Obrázek 10: Ukázka webové stránky

Každá mobilní síť je reprezentována jinou barvou. Pro GSM je použita barva zelená, pro WCDMA barva oranžová a síť LTE je zobrazena barvou modrou. Navíc jsou jednotlivé úrovně signálu odlišeny jiným odstínem, jak je znázorněno v tabulce 2.

Tabulka 2: Systém barevného značení

RSSI [dBm]				GSM	WCDMA	LTE
0	≥	-69	**Excellent**			
-70	≥	-99	**Fair**			
-100	≥	-120	**Poor**			

Při kliknutí na jakýkoliv ze zobrazených bodů na mapě se v pravé části stránky vypíší jeho detaily. Těmi jsou Date and Time, Device Latitude, Longitude, Altitude, Signal Strength, Network, PLMN, CellID, LAC, RSSI, Number of aggregated records a v závislosti na typu sítě také RSRP, RSCP, RSRQ, SINR a Ec/No. Strukturu detailů zvoleného bodu ukazuje obrázek 11.



Obrázek 11: Detaily zvoleného bodu

Pod mapou se dále nachází sada ovládácích prvků spolu s tabulkou. Ta zobrazuje pouze data agregovaných bodů právě viditelných na mapě. Přímo nad tabulkou se nachází tlačítko *"Export table to CSV"*, které, jak už název napovídá, slouží pro export dat viditelných bodů do CSV formátu. Exportovaná data mají strukturu a obsah tabulky Aggr, a jsou tedy bohatší o parametry z GPS a také z modemu.

Dále se v levé části nachází pole *"Device"* a *"Network"*. Ty nám dávají informaci o zařízeních a sítích obsažených ve zvoleném časovém horizontu, který je znázorněn údaji *"Time from"* a *"Time to"*, obrázek 12.

Obrázek 12: Ovládací prvky stránky

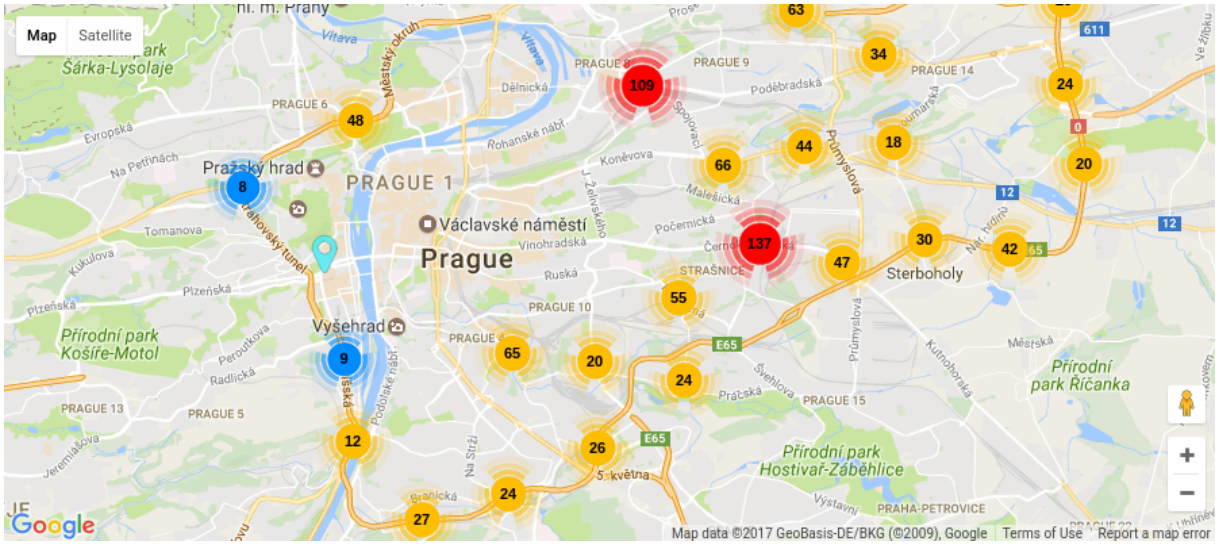
Pro volbu času pak slouží pole *"From"* a *"To"*. Zde se musí datum a čas vložit přesně ve stanoveném formátu, jinak dojde k nesprávnému přečtení a následnému zobrazení. Zvolený čas lze ještě dále manuálně upravit posuvníkem.

Pod údaji o čase se nachází dvojce tzv. "list boxů", umožňujících zvolení konkrétního zařízení nebo sítě, kterou chceme zobrazit. Možnost "All" zobrazí všechna dostupná zařízení nebo sítě.

Tlačítkem *"Show"* se na mapě zobrazí body odpovídající zvoleným parametrům. Tlačítko *"Export aggregated to CSV"* pak slouží k získání dat v CSV formátu. Rozdíl oproti *"Export table*

to CSV" je, že se exportují všechna data odpovídající zvoleným parametrům, nejen data právě viditelná na mapě.

Jak už bylo popsáno v sekci 4.3.5.1, body se při oddálení mapy shlukují do tzv. clusterů. Tím se do jisté míry zvýší přehlednost a odezva celého webu. Clustery jsou znázorněny na obrázku 13.



Obrázek 13: Shlukování bodů na mapě

5 Zhodnocení dosažených výsledků

V úvodních částech práce byly popsány základní struktury nepoužívanějších mobilních sítí dnešní doby, především pak sítí GSM, UMTS a LTE/EPC. Následně byly pro každou ze sítí uvedeny základní kvalitativní parametry signálu, které svými hodnotami vypovídají o úrovni signálu, jeho kvalitě či naopak chybovosti.

V praktické části práce byl navrhnut a realizován systém automatického měření parametrů signálu mobilní sítě. Ten se skládá z klienta a serveru. Klient je postaven na jednodeskovém počítači RaspberryPi, je vybaven GPS přijímačem, LTE modemem a WiFi adaptérem. Měřené parametry polohy a signálu jsou lokálně zpracovávány a odesílány na server přes HTTP spojení zabezpečené OpenVPN.

Na straně serveru jsou data přijímána a ukládána do databáze SQLite. Odtud se data z důvodu optimalizace rychlosti a přesnějších výsledků agregují pomocným skriptem a následně vizualizují na mapovém podkladu na webové stránce.

Sběr dat na klientské části vyžadoval práci s AT příkazy přímo na sériové lince modemu. Zde se zprvu projevovaly klasické problémy sériového rozhraní, například zahlcení linky apod. Tyto problémy byly odstraněny postupnými úpravami časování požadavků a ochrannými intervaly.

Lokální přístup se na klientskou část přes vlastní WiFi Ad-Hoc síť podařil zajistit. Síť však nepodporuje žádné zabezpečení a ovladače pro chod v Ad-Hoc módu se liší v závislosti na použitém Linuxovém jádru. Z tohoto důvodu toto řešení není doporučeno.

Co se týče serverové části, i přes značnou optimalizaci webového rozhraní je vykreslení více jak 5000 bodů na mapě velmi náročné. Je proto doporučeno volit časové rozpětí maximálně 10 dní. Jako dočasné řešení tohoto problému může být zvětšení časového rozestupu při sběru dat nebo zvětšení minimální vzdálenost, pro kterou se vykoná datová agregace.

I když cílem práce nebylo zmapování určité oblasti, z naměřených dat jsou zřejmá místa s horším pokrytím signálu, na které by se případně operátor Vodafone CZ mohl při optimalizaci své sítě zaměřit.

Zjištěné parametry v daném místě v mobilní síti ovlivňují úroveň poskytovaných služeb a to jak komfortu služeb datových, tak kvalitu řeči. V práci [15] a [16] byla zjišťována hodnota MOS (Mean Opinion Score) řeči v GSM síti pomocí metody PESQ (Perceptual Evaluation of Speech Quality), která je intrusivním objektivním způsobem hodnocení popsaným v doporučení ITU-P.862 [17], [18]. Jeden ze způsobů praktické PESQ implementace nástroje pro hodnocení kvality řeči je popsán v [19].

Při řešení byly splněny všechny body zadání. Mnou vytvořený nástroj dokáže v reálném čase mapovat parametry mobilního signálu a může být užtečný při optimalizaci pokrytí buňkových sítí. Při jednoduché změně dokáže vytvořený framework najít uplatnění i v anaýze pokrytí různých bezdrátových sítí.

Literatura

[1] HUURDEMAN, Anton A., *The worldwide history of telecommunications,* New York: J. Wiley, c2003. ISBN 0471205052.

[2] ZIGANGIROV, K. Sh., *Theory of code division multiple access communication,* Hoboken, NJ: Wiley, c2004. IEEE series on mobile & digital communication. ISBN 0471457124.

[3] EBERSPACHER, J., *GSM: architecture, protocols and services,* 3rd ed., English lang. ed. Chichester, U.K.: Wiley, c2009. ISBN 0470030704.

[4] MICHALEK, Libor a Roman ŠEBESTA, *Rádiové sítě II pro integrovanou výuku VUT a VŠB-TUO,* Ostrava: Vysoká škola báňská - Technická univerzita Ostrava, 2016. ISBN 9788024835594.

[5] DVORSKÝ, Marek a Jan TOMIS, *Radiokomunikační technika II: Návod na cvičení,* Ostrava, 2016. Dostupné také z: https://comtech.vsb.cz/moodle/

[6] RAPPAPORT, Theodore S., *Wireless communications: principles and practice,* 2nd ed. Upper Saddle River, N.J.: Prentice Hall PTR, c2002. ISBN 0130422320.

[7] *3GPP TS 36.214: LTE E-UTRA Physical layer - Measurements,* V9.1.0. Cadex: ETSI, 2010.

[8] *3GPP TS 36.133: LTE E-UTRA Requirements for support of radio resource management,* V8.2.0. Cadex: ETSI, 2008.

[9] *Fars Robotics Website: Installing the rtl8188eu based wifi adaptor driver for Raspbian,* Fars Robotics Website [online]. online: Fars Robotics, 2017 [cit. 2017-04-24]. Dostupné z: http://www.fars-robotics.net/

[10] *GitHubGist: python-gps,* GitHub [online]. San Francisco: GitHub, 2017 [cit. 2017-04-09]. Dostupné z: https://gist.github.com/wolfg1969/4653340

[11] *Haversine Formula in Python (Bearing and Distance between two GPS points),* Stack Overflow [online]. New York: Stack Overflow, 2011 [cit. 2017-04-17]. Dostupné z: http://stackoverflow.com/questions/4913349/haversine-formula-in-python-bearing-and-distance-between-two-gps-points

[12] *JQuery CDN: Latest Stable Versions,* JQuery [online]. online: jQuery, 2017 [cit. 2017-04-22]. Dostupné z: https://code.jquery.com/

[13] *Google Developers: Google Hosted Libraries,* Google Hosted Libraries [online]. Mountain View, California: Google, 2017 [cit. 2017-04-22]. Dostupné z: https://developers.google.com/speed/libraries/

[14] *GitHub: markerclustererplus,* GitHub [online]. San Francisco: GitHub, 2015 [cit. 2017-04-22]. Dostupné z: https://github.com/mahnunchik/markerclustererplus

[15] FAJKUS, M., M. MIKULEC, M. VOZNAK, M. TOMIS a P. FAZIO. *Speech quality measurement of GSM infrastructure built on USRP N210 and openBTS project,* Advances in Electrical and Electronic Engineering. 2014, 2014(4), 341-346.

[16] PARTILA, P., M. KOHUT, M. VOZNAK, M. MIKULEC, J. SAFARIK a K. TOMALA. *A methodology for measuring voice quality using PESQ and interactive voice response in the GSM channel designed by openBTS,* Advances in Electrical and Electronic Engineering. 2013, 2013(5), 380-386.

[17] RIX, A.W., M.P. HOLLIER, A.P. HEKSTRA a J.G. BEERENDS. *Perceptual evaluation of speech quality (PESQ): The new ITU standard for end-to-end speech quality assessment. Part I - Time-delay compensation,* AES: Journal of the Audio Engineering Society. 2002, 2002(10), 755-764.

[18] BEERENDS, J.G., A.P. HEKSTRA, A.W. RIX a M.P. HOLLIER. *Perceptual evaluation of speech quality (PESQ): The new ITU standard for end-to-end speech quality assessment. Part II - Psychoacoustic model.* AES: Journal of the Audio Engineering Society. 2002, 2002(10), 765-778.

[19] ROZHON, J. a M. VOZNAK. *Development of a speech quality monitoring tool based on ITU-T P.862.* 2011 34th International Conference on Telecommunications and Signal Processing, TSP 2011 - Proceedings. 2011, 2011(6043771), 62-66.

A Seznam instalovaných balíků - klient

Seznam instalovaných balíků je dostupný také v datové příloze, soubor `packages-client.txt`.

acl
adduser
alsa-utils
apt
apt-listchanges
apt-utils
aptitude
aptitude-common
avahi-daemon
base-files
base-passwd
bash
bash-completion
bind9-host
binutils
bluez
bluez-firmware
bsdmainutils
bsdutils
build-essential
bzip2
ca-certificates
cifs-utils
comgt
console-setup
console-setup-linux
coreutils
cpio
cpp
cpp-4.9
crda
cron
curl
dash
dbus
dc
debconf
debconf-i18n
debconf-utils
debianutils
device-tree-compiler
dhcpcd5
diffutils
dmidecode
dmsetup
dosfstools
dphys-swapfile
dpkg
dpkg-dev
e2fslibs
e2fsprogs
easy-rsa
ed
fake-hwclock
fakeroot
fbset
file
findutils
firmware-atheros
firmware-brcm80211
firmware-libertas
firmware-ralink
firmware-realtek
g++
g++-4.9
gcc
gcc-4.6-base
gcc-4.7-base
gcc-4.8-base
gcc-4.9
gcc-4.9-base
gcom
gdb
gdbserver
geoip-database
gnupg
gpgv
gpsd
grep
groff-base
gzip
hardlink
hostname
ifupdown
info
init
init-system-helpers
initramfs-tools
initscripts
insserv
install-info
iproute2
iptables
iputils-ping
isc-dhcp-client
isc-dhcp-common
isc-dhcp-server
iso-codes
iw
kbd
keyboard-configuration
klibc-utils
kmod
less
libacl1

libalgorithm-c3-perl
libalgorithm-diff-perl
libalgorithm-diff-xs-perl
libalgorithm-merge-perl
libapt-inst1.5
libapt-pkg4.12
libarchive-extract-perl
libasan1
libasound2
libasound2-data
libatomic1
libattr1
libaudit-common
libaudit1
libavahi-common-data
libavahi-common3
libavahi-core7
libbind9-90
libblkid1
libbluetooth3
libboost-iostreams1.49.0
libboost-iostreams1.50.0
libboost-iostreams1.53.0
libboost-iostreams1.54.0
libboost-iostreams1.55.0
libbsd0
libbz2-1.0
libc-bin
libc-dev-bin
libc6
libc6-dbg
libc6-dev
libcap-ng0
libcap2
libcap2-bin
libcgi-fast-perl
libcgi-pm-perl
libclass-c3-perl
libclass-c3-xs-perl
libcloog-isl4
libcomerr2
libcpan-meta-perl
libcryptsetup4
libcurl3
libcwidget3
libdaemon0
libdata-optlist-perl
libdata-section-perl
libdb5.3
libdbus-1-3
libdebconfclient0
libdevmapper1.02.1
libdns-export100
libdns100
libdpkg-perl
libdrm2
libedit2
libestr0
libevent-2.0-5
libexpat1
libfakeroot
libfcgi-perl
libffi6
libfile-fcntllock-perl
libfreetype6
libfreetype6-dev
libgcc-4.9-dev
libgcc1
libgcrypt20
libgdbm3
libgeoip1
libglib2.0-0
libglib2.0-data
libgmp10
libgnutls-deb0-28
libgnutls-openssl27
libgomp1
libgpg-error0
libgpm2
libgps21
libgssapi-krb5-2
libhogweed2
libicu52
libident
libidn11
libio-pty-perl
libipc-run-perl
libirs-export91
libisc-export95
libisc95
libisccc90
libisccfg-export90
libisccfg90
libisl10
libiw30
libjim0.75
libjpeg62-turbo
libjson-c2
libk5crypto3
libkeyutils1
libklibc
libkmod2
libkrb5-3
libkrb5support0
libldap-2.4-2
liblocale-gettext-perl
liblog-message-perl
liblog-message-simple-perl
liblogging-stdlog0
liblognorm1
libluajit-5.1-common
liblwres90
liblzma5
liblzo2-2
libmagic1
libmodule-build-perl
libmodule-pluggable-perl

libmodule-signature-perl
libmount1
libmpc3
libmpfr4
libmro-compat-perl
libncurses5
libncursesw5
libnettle4
libnewt0.52
libnfnetlink0
libnfsidmap2
libnih-dbus1
libnih1
libnl-3-200
libnl-genl-3-200
libnss-mdns
libopts25
libp11-kit0
libpackage-constants-perl
libpam-modules
libpam-modules-bin
libpam-runtime
libpam0g
libparams-util-perl
libparted2
libpcap0.8
libpci3
libpcre3
libpcsclite1
libpipeline1
libpkcs11-helper1
libplymouth4
libpng12-0
libpng12-dev
libpod-latex-perl
libpod-readme-perl
libpopt0
libprocps3
libpsl0
libpython-stdlib
libpython2.7
libpython2.7-minimal
libpython2.7-stdlib
libqmi-glib1
libraspberrypi-bin
libraspberrypi-dev
libraspberrypi-doc
libraspberrypi0
libreadline6
libregexp-common-perl
librtmp1
libsamplerate0
libsasl2-2
libsasl2-modules
libsasl2-modules-db
libselinux1
libsemanage-common
libsemanage1
libsepol1
libsigc++-1.2-5c2
libsigc++-2.0-0c2a
libslang2
libsmartcols1
libsoftware-license-perl
libsqlite3-0
libss2
libssh2-1
libssl1.0.0
libstdc++-4.9-dev
libstdc++6
libsub-exporter-perl
libsub-install-perl
libsysfs2
libsystemd0
libtalloc2
libtasn1-6
libterm-ui-perl
libtext-charwidth-perl
libtext-iconv-perl
libtext-soundex-perl
libtext-template-perl
libtext-wrapi18n-perl
libtimedate-perl
libtinfo5
libtirpc1
libubsan0
libudev0
libudev1
libusb-0.1-4
libusb-1.0-0
libustr-1.0-1
libuuid1
libv4l-0
libv4l2rds0
libv4lconvert0
libwbclient0
libwrap0
libx11-6
libx11-data
libxapian22
libxau6
libxcb1
libxdmcp6
libxext6
libxml2
libxmuu1
libxtables10
linux-libc-dev
locales
login
logrotate
lsb-base
lsb-release
lshw
lua5.1
luajit
make

makedev
man-db
manpages
manpages-dev
mawk
mime-support
module-init-tools
moreutils
mount
mountall
multiarch-support
nano
ncdu
ncurses-base
ncurses-bin
ncurses-term
net-tools
netbase
netcat-openbsd
netcat-traditional
nfs-common
ntp
openresolv
opensc
opensc-pkcs11
openssh-client
openssh-server
openssh-sftp-server
openssl
openvpn
parted
passwd
patch
pciutils
perl
perl-base
perl-modules
pi-bluetooth
pkg-config
plymouth
ppp
procps
psmisc
python
python-apt
python-apt-common
python-gps
python-minimal
python-rpi.gpio
python-serial
python-support
python2.7
python2.7-minimal
raspberrypi-bootloader
raspberrypi-kernel
raspberrypi-net-mods
raspberrypi-sys-mods
raspbian-archive-keyring
raspi-config
raspi-copies-and-fills
readline-common
rename
rpcbind
rsyslog
samba-common
screen
sed
sensible-utils
sgml-base
shared-mime-info
ssh
startpar
strace
sudo
systemd
systemd-sysv
sysv-rc
sysvinit-utils
tar
tasksel
tasksel-data
tcpd
telnet
traceroute
triggerhappy
tzdata
ucf
udev
unzip
usb-modeswitch
usb-modeswitch-data
usbutils
util-linux
v4l-utils
vim
vim-common
vim-runtime
vim-tiny
wget
whiptail
whois
wireless-regdb
wireless-tools
wpasupplicant
xauth
xdg-user-dirs
xkb-data
xml-core
xz-utils
zlib1g
zlib1g-dev

B Seznam instalovaných balíků - server

Seznam instalovaných balíků je dostupný také v datové příloze, soubor `packages-server.txt`.

accountsservice
acpid
adduser
apache2
apache2-bin
apache2-data
apache2-utils
apparmor
apport
apport-symptoms
apt
apt-transport-https
apt-utils
at
autoconf
automake
autotools-dev
base-files
base-passwd
bash
bash-completion
bc
bind9-host
binutils
biosdevname
bsdmainutils
bsdutils
build-essential
busybox-initramfs
busybox-static
byobu
bzip2
ca-certificates
cgmanager
command-not-found
command-not-found-data
console-setup
console-setup-linux
coreutils
cpio
cpp
cpp-5
crda
cron
curl
dash
dbus
debconf
debconf-i18n
debhelper
debianutils
dh-php
dh-python
dh-strip-nondeterminism
dictionaries-common
diffutils
discover
discover-data
distro-info-data
dmidecode
dmsetup
dnsutils
dosfstools
dpkg
dpkg-dev
e2fslibs
e2fsprogs
easy-rsa
ed
eject
emacsen-common
ethtool
fakeroot
file
findutils
fonts-ubuntu-font-family-console
friendly-recovery
ftp
fuse
g++
g++-5
gawk
gcc
gcc-4.8-base
gcc-4.9-base
gcc-5
gcc-5-base
gcc-6-base
geoip-database
gettext
gettext-base
gir1.2-glib-2.0
gnupg
gpgv
grep
groff-base
grub-common
grub-gfxpayload-lists
grub-pc
grub-pc-bin
grub2-common
gzip
hdparm
hostname

ifupdown
info
init
init-system-helpers
initramfs-tools
initramfs-tools-bin
initramfs-tools-core
initscripts
insserv
install-info
installation-report
intltool-debian
iproute2
iptables
iputils-ping
iputils-tracepath
irqbalance
isc-dhcp-client
isc-dhcp-common
iso-codes
iw
kbd
keyboard-configuration
klibc-utils
kmod
krb5-locales
landscape-common
language-pack-en
language-pack-en-base
language-pack-gnome-en
language-pack-gnome-en-base
language-selector-common
laptop-detect
less
libaccountsservice0
libacl1
libaio1
libalgorithm-diff-perl
libalgorithm-diff-xs-perl
libalgorithm-merge-perl
libapache2-mod-php
libapache2-mod-php7.0
libapparmor-perl
libapparmor1
libapr1
libaprutil1
libaprutil1-dbd-sqlite3
libaprutil1-ldap
libapt-inst1.5
libapt-inst2.0
libapt-pkg4.12
libapt-pkg5.0
libarchive-extract-perl
libarchive-zip-perl
libasan2
libasn1-8-heimdal
libasprintf-dev
libasprintf0v5
libatomic1
libattr1
libaudit-common
libaudit1
libbind9-140
libbind9-90
libblkid1
libboost-iostreams1.58.0
libbsd0
libbz2-1.0
libc-bin
libc-dev-bin
libc6
libc6-dev
libcap-ng0
libcap2
libcap2-bin
libcc1-0
libccid
libcgi-fast-perl
libcgi-pm-perl
libcgmanager0
libcilkrts5
libck-connector0
libcomerr2
libcroco3
libcryptsetup4
libcurl3
libcurl3-gnutls
libdb5.3
libdbus-1-3
libdbus-glib-1-2
libdebconfclient0
libdevmapper1.02.1
libdiscover2
libdns-export162
libdns100
libdns162
libdpkg-perl
libdrm2
libedit2
libelf1
libencode-locale-perl
libestr0
libevent-2.0-5
libevent-core-2.0-5
libexpat1
libexporter-tiny-perl
libfakeroot
libfcgi-perl
libfdisk1
libffi6
libfile-fcntllock-perl
libfile-stripnondeterminism-perl
libfreetype6
libfribidi0
libfuse2
libgc1c2

libgcc-5-dev
libgcc1
libgck-1-0
libgcr-3-common
libgcr-base-3-1
libgcrypt11
libgcrypt20
libgdbm3
libgeoip1
libgettextpo-dev
libgettextpo0
libgirepository-1.0-1
libglib2.0-0
libglib2.0-data
libgmp10
libgnutls-openssl27
libgnutls26
libgnutls30
libgomp1
libgpg-error0
libgpm2
libgssapi-krb5-2
libgssapi3-heimdal
libhcrypto4-heimdal
libheimbase1-heimdal
libheimntlm0-heimdal
libhogweed4
libhtml-parser-perl
libhtml-tagset-perl
libhtml-template-perl
libhttp-date-perl
libhttp-message-perl
libhx509-5-heimdal
libicu55
libidn11
libio-html-perl
libisc-export160
libisc160
libisc95
libisccc140
libisccc90
libisccfg140
libisccfg90
libisl15
libitm1
libiw30
libjson-c2
libjson0
libk5crypto3
libkeyutils1
libklibc
libkmod2
libkrb5-26-heimdal
libkrb5-3
libkrb5support0
libldap-2.4-2
liblist-moreutils-perl
liblocale-gettext-perl
liblockfile-bin
liblockfile1
liblog-message-perl
liblog-message-simple-perl
liblsan0
libltdl-dev
libltdl7
liblua5.1-0
liblwp-mediatypes-perl
liblwres141
liblwres90
liblz4-1
liblzma5
liblzo2-2
libmagic1
libmail-sendmail-perl
libmcrypt4
libmnl0
libmodule-pluggable-perl
libmodule-runtime-perl
libmount1
libmpc3
libmpdec2
libmpfr4
libmpx0
libncurses5
libncursesw5
libnettle6
libnewt0.52
libnfnetlink0
libnih-dbus1
libnih1
libnl-3-200
libnl-genl-3-200
libnuma1
libp11-kit0
libpam-cap
libpam-modules
libpam-modules-bin
libpam-runtime
libpam-systemd
libpam0g
libparams-classify-perl
libparted0debian1
libparted2
libpcap0.8
libpci3
libpcre16-3
libpcre3
libpcre3-dev
libpcre32-3
libpcrecpp0v5
libpcsclite1
libperl5.22
libpipeline1
libpkcs11-helper1
libplymouth2
libplymouth4
libpng12-0

libpod-latex-perl
libpolkit-agent-1-0
libpolkit-backend-1-0
libpolkit-gobject-1-0
libpopt0
libprocps3
libprocps4
libpython-stdlib
libpython2.7
libpython2.7-minimal
libpython2.7-stdlib
libpython3-stdlib
libpython3.5
libpython3.5-minimal
libpython3.5-stdlib
libquadmath0
libreadline5
libreadline6
libroken18-heimdal
librtmp0
librtmp1
libsasl2-2
libsasl2-modules
libsasl2-modules-db
libseccomp2
libselinux1
libsemanage-common
libsemanage1
libsepol1
libsigsegv2
libslang2
libsmartcols1
libsqlite0
libsqlite3-0
libsqlite3-dev
libss2
libssl-dev
libssl-doc
libssl1.0.0
libstdc++-5-dev
libstdc++6
libsys-hostname-long-perl
libsystemd-daemon0
libsystemd-login0
libsystemd0
libtasn1-6
libterm-ui-perl
libtext-charwidth-perl
libtext-iconv-perl
libtext-soundex-perl
libtext-wrapi18n-perl
libtimedate-perl
libtinfo5
libtool
libtsan0
libubsan0
libudev1
libunistring0
liburi-perl
libusb-0.1-4
libusb-1.0-0
libustr-1.0-1
libutempter0
libuuid1
libwind0-heimdal
libwrap0
libx11-6
libx11-data
libxapian-1.3-5
libxapian22v5
libxau6
libxcb1
libxdmcp6
libxext6
libxml2
libxmuu1
libxslt1.1
libxtables10
libxtables11
linux-base
linux-firmware
linux-generic
linux-headers-4.4.0-57
linux-headers-4.4.0-57-generic
linux-headers-4.4.0-66
linux-headers-4.4.0-66-generic
linux-headers-4.4.0-70
linux-headers-4.4.0-70-generic
linux-headers-4.4.0-71
linux-headers-4.4.0-71-generic
linux-headers-4.4.0-72
linux-headers-4.4.0-72-generic
linux-headers-generic
linux-image-4.4.0-57-generic
linux-image-4.4.0-66-generic
linux-image-4.4.0-70-generic
linux-image-4.4.0-71-generic
linux-image-4.4.0-72-generic
linux-image-extra-4.4.0-57-generic
linux-image-extra-4.4.0-66-generic
linux-image-extra-4.4.0-70-generic
linux-image-extra-4.4.0-71-generic
linux-image-extra-4.4.0-72-generic
linux-image-generic
linux-libc-dev
locales
lockfile-progs
login
logrotate
lsb-base
lsb-release
lshw

lsof
ltrace
m4
make
makedev
man-db
manpages
manpages-dev
mawk
memtest86+
mime-support
mlocate
module-init-tools
mount
mountall
mtr-tiny
multiarch-support
mysql-client-5.7
mysql-client-core-5.7
mysql-common
mysql-server
mysql-server-5.7
mysql-server-core-5.7
nano
ncurses-base
ncurses-bin
ncurses-term
net-tools
netbase
netcat-openbsd
ntfs-3g
ntpdate
opensc
opensc-pkcs11
openssh-client
openssh-server
openssh-sftp-server
openssl
openvpn
os-prober
parted
passwd
pastebinit
patch
pciutils
pcscd
perl
perl-base
perl-modules-5.22
php
php-common
php-mcrypt
php-pear
php-xml
php7.0
php7.0-cli
php7.0-common
php7.0-dev
php7.0-fpm
php7.0-json
php7.0-mcrypt
php7.0-opcache
php7.0-readline
php7.0-sqlite3
php7.0-xml
pkg-php-tools
plymouth
plymouth-theme-ubuntu-text
po-debconf
policykit-1
popularity-contest
powermgmt-base
ppp
pppconfig
pppoeconf
procps
psmisc
python
python-apt
python-apt-common
python-attr
python-cffi-backend
python-chardet
python-configobj
python-cryptography
python-debian
python-enum34
python-gdbm
python-idna
python-ipaddress
python-minimal
python-ndg-httpsclient
python-openssl
python-pam
python-pkg-resources
python-pyasn1
python-pyasn1-modules
python-requests
python-serial
python-service-identity
python-six
python-twisted-bin
python-twisted-core
python-urllib3
python-zope.interface
python2.7
python2.7-minimal
python3
python3-apport
python3-apt
python3-chardet
python3-commandnotfound
python3-dbus
python3-debian
python3-distupgrade
python3-gdbm
python3-gi

python3-minimal
python3-newt
python3-pkg-resources
python3-problem-report
python3-pycurl
python3-requests
python3-six
python3-software-properties
python3-systemd
python3-update-manager
python3-urllib3
python3-xapian1.3
python3.5
python3.5-minimal
readline-common
rename
resolvconf
rsync
rsyslog
run-one
screen
sed
sensible-utils
sgml-base
shared-mime-info
shtool
software-properties-common
sqlite
sqlite3
ssh-import-id
ssl-cert
strace
sudo
systemd
systemd-shim
systemd-sysv
sysv-rc
sysvinit-utils
tar
tasksel
tasksel-data
tcpd
tcpdump
telnet
thermald
time
tmux
tzdata
ubuntu-keyring
ubuntu-minimal
ubuntu-release-upgrader-core
ubuntu-standard
ucf
udev
ufw
unattended-upgrades
update-manager-core
update-motd
update-notifier-common
upstart
ureadahead
usbutils
util-linux
uuid-runtime
vim
vim-common
vim-runtime
vim-tiny
w3m
wamerican
wbritish
wget
whiptail
wireless-regdb
wireless-tools
wpasupplicant
xauth
xdg-user-dirs
xkb-data
xml-core
xz-utils
zlib1g
zlib1g-dev

C Zdrojový kód souboru modem-client.py

Zdrojový kód je dostupný také v datové příloze této práce, v souboru `modem-client.py`.

```
#!/usr/bin/env python2
# Written by Martin Talas March 2017
# License: GPL 2.0

import os
from gps import *
import time
import threading
import datetime
import io
import sys
import subprocess
import re
import fcntl
import serial
import urllib
import urllib2
import csv
import httplib

## Register in network
# "4G","3G","2G", "AUTO"
register = None
config_file = "/home/pi/register.conf"

# Global variables
gpsd = None
modem = None
csq = ''
plmn = ''
creg = ''
gsm_hcsq = ''
umts_hcsq = ''
lte_hcsq = ''
lte_hfreqinfo = ''
network = None
sending_enabled = 0
url = 'http://10.8.0.1:8005' # IP/port HTTP server is reachable on
base = {}


#fieldnames = ("timestamp","device","lat","lon","alt","eps","epx","epv","ept","
    speed","csq","ber",
#     "plmn_cc","plmn_nc","network","rssi","rsrp_rscp","sinr_ecio","rsrq","
    reg_mode",
#     "reg_stat","lac","cellid","zero","six","lte_band","earfcn_dl","freq_dl","
    bw_dl",
#     "earfcn_up","freq_up","bw_up","aggr_num")

class GpsPoller(threading.Thread):
   def __init__(self):
      threading.Thread.__init__(self)
      global gpsd #bring it in scope
      gpsd = gps(mode=WATCH_ENABLE) #starting the stream of info
```

```
        self.current_value = None
        self.running = True #setting the thread running to true
        self.modem_running = True #setting the thread running to true

    def run(self):
        global gpsd
        while gpsp.running:
            gpsd.next() #this will continue to loop and grab EACH set of gpsd info
                  #to clear the buffer

def checkConfig():
    if os.path.exists(config_file) == False:
        f = open(config_file, 'w')
        f.write("AUTO")
        f.close()
        return "AUTO"
    else:
        with open(config_file, 'r') as f:
            config = f.readline().rstrip()
        return config

def internetConnectionCheck():
    try:
        urllib2.urlopen('http://idnes.cz', timeout=1) #idnes.cz
        return True
    except urllib2.URLError as err:
        return False

def getDeviceId():
    with open('/sys/class/net/eth0/address') as f:
        mac = f.read().strip()
    return mac.translate(None, ':')

def parseValues(list):
    try:
        for l in list:
            if re.search( r'[+]CSQ:', l, re.M|re.I):
                output = re.match(r'\+CSQ: ([0-9]{1,2}),([0-9]{2})\r', l, re.M|re.I)
                #print "re.csq:", l
                return output.groups()
                #return l[6:-2]
            if re.search( r'HCSQ:', l, re.M|re.I):
                output = re.match(r'\^HCSQ:\"([A-Z]*)\",([0-9]{1,3})
                    (?:\,([0-9]{0,3}))?(?:\,([0-9]{0,3}))?(?:\,([0-9]{0,3}))?', l,
                    re.M|re.I)
                #print "re.hcsq:", l
                return output.groups()
                #return l[6:-2]
            if re.search( r'PLMN', l, re.M|re.I):
                output = re.match(r'\^PLMN: ([0-9]{3}),([0-9]{2})', l, re.M|re.I)
                #print "re.plmn:", l
                return output.groups()
                #return l[7:-2]
            if re.search( r'CREG', l, re.M|re.I):
                output = re.match(r'\+CREG: ([0-9]{1}),([0-9]{1})(?:\,*"([A-Z0-9]*)
                    ","([A-Z0-9]*)")?', l, re.M|re.I)
                #print "re.creg:", l
                return output.groups()
                #return l[7:-2]
```

```
            if re.search( r'HFREQINFO', l, re.M|re.I):
               output = re.search(r'\^HFREQINFO:([0-9]*),([0-9]*),([0-9]*),([0-9]*)
                   ,([0-9]*),([0-9]*),([0-9]*),([0-9]*),([0-9]*)', l, re.M|re.I)
               #print "re.hfreqinfo:", l
               return output.groups()
               #return l[11:-2]
            if re.search(r'SYSCFGEX', l, re.M|re.I):
               output = re.match(r'\^SYSCFGEX:\"([0-9]*)\"', l, re.M|re.I)
               #print "re.syscfgex:", l
               return output.groups()
            #else:
               #  return "Hello"
      except AttributeError:
         pass


def connectToInternet():
      global gpsp
      time.sleep(10)
      while gpsp.running:
         if internetConnectionCheck() == False:
            print "Internet connection is down!"
            subprocess.call(["sudo", "killall", "dhclient"])
            subprocess.call(["sudo", "sh", "/home/pi/connect.sh"])
         else:
            print "Internet connection is up!"
         time.sleep(20)


def registerNetwork():
      global modem, register
      register = checkConfig()
      try:
         modem.write("AT+CREG=2\r")
         time.sleep(0.2)
         modem.write("AT^SYSCFGEX?\r")
         syscfg = parseValues(modem.readlines(64))
         #print syscfg
         syscfg = str(syscfg[0])
         #print syscfg
         time.sleep(1)
         if register == "4G":
            if syscfg != "03":
               modem.write("AT^SYSCFGEX=\"03\",3fffffff,2,4,7FFFFFFFFFFFFFFF,,\r")
               time.sleep(4)
               print "This is 4G!"
         elif register == "3G":
            if syscfg != "02":
               modem.write("AT^SYSCFGEX=\"02\",3fffffff,2,4,7FFFFFFFFFFFFFFF,,\r")
               time.sleep(5)
               print "This is 3G!"
         elif register == "2G":
            if syscfg != "01":
               modem.write("AT^SYSCFGEX=\"01\",3fffffff,2,4,7FFFFFFFFFFFFFFF,,\r")
               time.sleep(4)
               print "This is 2G!"
         elif syscfg != "00":
            modem.write("AT^SYSCFGEX=\"00\",3fffffff,2,4,7FFFFFFFFFFFFFFF,,\r")
            time.sleep(4)
```

```
            print "This is AUTO!"
        except TypeError as err:
            print "Error during registration:", err



def modemData():
    global modem, register, network, csq, plmn, lte_hcsq, umts_hcsq, gsm_hcsq,
        creg, lte_hfreqinfo, gpsp, sending_enabled
    syscfg = None
    try:
        modem = serial.Serial(port='/dev/ttyUSB1', baudrate=115200, bytesize=8,
            parity='N', stopbits=1, timeout=0.6)
        while gpsp.modem_running:
            registerNetwork()
            try:
                modem.write("AT^HCSQ?\r")
                response = modem.readlines(40)
                #print response
                if len([k for k in response if "LTE" in k]) > 0:
                    network = "LTE"
                    print network
                    lte_hcsq_tmp = parseValues(response)
                    if lte_hcsq_tmp != None and len(lte_hcsq_tmp) == 5:
                        lte_hcsq_tmp = list(lte_hcsq_tmp)
                        #print lte_hcsq_tmp
                        lte_hcsq_tmp[1] = str(int(lte_hcsq_tmp[1]) - 120)
                        lte_hcsq_tmp[2] = str(int(lte_hcsq_tmp[2]) - 140)
                        lte_hcsq_tmp[3] = str(round(0.2 * int(lte_hcsq_tmp[3]) - 20,1)
                            )
                        lte_hcsq_tmp[4] = str(round(0.5 * int(lte_hcsq_tmp[4]) -
                            19.5,1))
                        lte_hcsq = lte_hcsq_tmp
                    else:
                        lte_hcsq = ['LTE','','','','']
                    #print "lte_hcqs:", lte_hcsq
                    modem.write("AT^HFREQINFO?\r")
                    hfreqinfo_tmp = parseValues(modem.readlines(64))
                    if hfreqinfo_tmp != None and len(hfreqinfo_tmp) == 9:
                        lte_hfreqinfo = list(hfreqinfo_tmp)
                    else:
                        lte_hfreqinfo = ['','','','','','','','','']
                    #print "lte_hfreqinfo:", lte_hfreqinfo
                elif len([k for k in response if "WCDMA" in k]) > 0:
                    network = "WCDMA"
                    print network
                    hcsq_tmp = parseValues(response)
                    if hcsq_tmp != None and len(hcsq_tmp) == 5:
                        hcsq_tmp = list(hcsq_tmp)
                        #print hcsq_tmp
                        hcsq_tmp[1] = str(int(hcsq_tmp[1]) - 120)
                        hcsq_tmp[2] = str(int(hcsq_tmp[2]) - 120)
                        hcsq_tmp[3] = str(round(0.5*int(hcsq_tmp[3]) - 32,1))
                        #hcsq_tmp.extend([''])
                        umts_hcsq = hcsq_tmp
                    else:
                        umts_hcsq = ['WCDMA','','','','']
                    #print "umts_hcsq:", umts_hcsq
```

```
            elif len([k for k in response if "GSM" in k]) > 0:
               network = "GSM"
               print network
               hcsq_tmp = parseValues(response)
               if hcsq_tmp != None and len(hcsq_tmp) == 5:
                  hcsq_tmp = list(hcsq_tmp)
                  #print hcsq_tmp
                  hcsq_tmp[1] = str(int(hcsq_tmp[1]) - 120)
                  #hcsq_tmp.extend(['','',''])
                  gsm_hcsq = hcsq_tmp
               else:
                  gsm_hcsq = ['GSM','','','','']
               print gsm_hcsq
            else:
               network = "None"
               time.sleep(5)

            modem.write("AT+CREG?\r")
            creg_tmp = parseValues(modem.readlines(64))
            if creg_tmp != None and len(creg_tmp) == 4:
               creg = list(creg_tmp)
            else:
               creg = ['','','','']
            #print "creg:", creg
            modem.write("AT+CSQ\r")
            csq_tmp = parseValues(modem.readlines(64))
            if csq_tmp != None and len(csq_tmp) == 2:
               csq = list(csq_tmp)
            else:
               csq = ['','']
            #print "csq:", csq
            modem.write("AT^PLMN?\r")
            plmn_tmp = parseValues(modem.readlines(64))
            if plmn_tmp != None and len(plmn_tmp) == 2:
               plmn = list(plmn_tmp)
            else:
               plmn = ['','']
            #print "plmn:", plmn
            sending_enabled = 1

         except (TypeError) as e:
            print "Exception:", e
      except serial.SerialException as err:
         print "Error:", err

   def sendData():
      global base, url, gpsp, sending_enabled
      while sending_enabled != 1:
         print "Sender is waiting for modem..."
         time.sleep(5)
      print "Launching Sender loop...."
      #time.sleep(10)
      while gpsp.running:
         try:
            key = base.keys()[0]
            #print "Key is: ", key
            base_data = base.get(key)
            #print "base_data:", base_data
            data = {'transaction-id':str(base_data[0]),'data':str(base_data[1:])}
```

```
            print data["transaction-id"], "sending..."
            params = urllib.urlencode(data)
            post_req = urllib2.Request(url, params)
            post_req.add_data(params)

            response = urllib2.urlopen(post_req)
            res_code = response.getcode()
            if res_code == 200:
               try:
                  print data["transaction-id"], "200 OK"
                  del base[key]
               except KeyError as e:
                  print "Error while deleting", data["transaction-id"], e
               #response_data = response.read()
               #response.close()
            else:
               print data["transaction-id"], res_code
            res_code = 0
         except (httplib.BadStatusLine, urllib2.URLError) as e:
            print "Error during sending:", e
            time.sleep(10)
            continue
         except (IndexError):
            print "Base is empty, nothing to send."
            time.sleep(20)
            continue



if __name__ == '__main__':
   #time.sleep(15)
   gpsp = GpsPoller() # create the thread
   register = checkConfig()
   print "Chosen network:", register
   deviceid = getDeviceId()
   print "Device ID:", deviceid
   #time.sleep(10)
   modemp = threading.Thread(target=modemData, name="T2-MODEM")
   transfer = threading.Thread(target=sendData, name="T3-SENDER")
   connection = threading.Thread(target=connectToInternet, name="T4-CONNECT")
   #modemData()
   print "Starting threads!"
   try:
      gpsp.start() # start it up
      modemp.start()
      transfer.start()
      connection.start()
      time.sleep(10)  # This times MUST last longer than first modem cycle
      while True:
         timestamp = '{:%Y%m%d%H%M%S}'.format(datetime.datetime.now())
         if network == "LTE":
            hcsq = lte_hcsq
            hfreqinfo = lte_hfreqinfo
         elif network == "WCDMA":
            hcsq = umts_hcsq
            hcsq[4] = ''
            hfreqinfo = ['','','','','','','','','']
         elif network == "GSM":
            hcsq = gsm_hcsq
```

```
            try:
               for n in range(2,5):
                  hcsq[n] = ''
            except TypeError:
               continue
            hfreqinfo = ['','','','','','','','','']
         else:
            continue

         try:
            print(threading.enumerate())
            data = [timestamp, deviceid, gpsd.fix.latitude, gpsd.fix.longitude,
                  gpsd.fix.altitude, gpsd.fix.eps, gpsd.fix.epx, gpsd.fix.epv,
                  gpsd.fix.ept, gpsd.fix.speed] + csq + plmn + hcsq + creg +
                     hfreqinfo
            data.append('0')
            print "Storing", data[0], ":", data[1:]

            if not (str(data[2]) == "nan" or str(data[2]) == "0.0" or data[3] ==
                "nan" or str(data[3]) == "0.0"):
               base[timestamp] = map(str,data)
            time.sleep(2)
         except TypeError as err:
            print err
            continue

   except (KeyboardInterrupt, SystemExit): #when you press ctrl+c
      print "\nKilling Thread..."
      gpsp.running = False
      gpsp.modem_running = False
      gpsp.join() # wait for the thread to finish what it's doing
      modemp.join()
      transfer.join()
      print "Number of items left in base:", len(base)
      if len(base) > 0:
         print "Items:", base
print "Done.\nExiting."
```

Výpis 24: Kompletní výpis souboru modem-client.py

D Seznam použitých AT příkazů

```
AT\^NDISDUP=1,1,\"internet\"   # Registrace do datove site
AT^HCSQ?              # Vypise parametry RSSI,RSRP,RSRQ,SINR,RSCP,ECNO
                      # Zavisi na typu site
AT^HFREQINFO?         # Vypise pouzite frekvence, pasma a EARFCN (Pouze pro LTE)
AT+CREG?              # Vypise mj. LAC a CellID
AT+CSQ                # Vypise hodnotu CSQ a BER
AT^PLMN?              # Vypise CC a NC dane site

AT^SYSCFGEX?          # Zjisti aktualni registraci v siti a mod provozu
AT^SYSCFGEX=\"03\",3fffffff,2,4,7FFFFFFFFFFFFFFF,, # Mod 4G
AT^SYSCFGEX=\"02\",3fffffff,2,4,7FFFFFFFFFFFFFFF,, # Mod 3G
AT^SYSCFGEX=\"01\",3fffffff,2,4,7FFFFFFFFFFFFFFF,, # Mod 2G
AT^SYSCFGEX=\"00\",3fffffff,2,4,7FFFFFFFFFFFFFFF,,  # Mod AUTO
```

Výpis 25: Seznam všech použitých AT příkazů

E Zdrojový kód souboru modem-server.py

Zdrojový kód je dostupný také v datové příloze této práce, v souboru `modem-server.py`.

```
#!/usr/bin/env python2
# -*- coding: utf-8 -*

'''
    http-serve [OPTIONS]
            --debug
            --verbose
            --db <name of the output database file>
            --log-file <logging file name>
            --port
'''

import os
import sys
import getopt
import logging
import sqlite3 as lite
import BaseHTTPServer
import urlparse

try:
    from cStringIO import StringIO
except ImportError:
    from StringIO import StringIO

'''
Create logger
'''
try:
    logging.captureWarnings(True)
except:
    pass

log = logging.getLogger(os.path.basename(sys.argv[0]))
log.setLevel(logging.ERROR)

ch = logging.StreamHandler()
ch.setLevel(logging.DEBUG)
ch_formatter = logging.Formatter("%(asctime)s - %(name)s - %(levelname)s - %(
    message)s")
ch.setFormatter(ch_formatter)
log.addHandler(ch)



LC = None

class dataObject:
    recId = -1
    transactionId = unicode('<unassigned>')
    #timestamp = unicode('<unassigned>')
    values = unicode('<unassigned>')
    @staticmethod
    def createSQLTable(lc):
```

```
        cur = lc.cursor()
        try:
            log.info("Create SQL Database Table Data")
            #cur.execute("DROP TABLE IF EXISTS Data")
            cur.execute("CREATE TABLE IF NOT EXISTS Data(timestamp TEXT NOT NULL
                ,"\
                        "device TEXT NOT NULL,lat TEXT NOT NULL,lon TEXT NOT NULL
                            ,alt TEXT,"\
                        "speed TEXT,epx TEXT,epy TEXT,epv TEXT,eps TEXT,csq TEXT
                            NOT NULL,"\
                        "ber TEXT,plmn_cc TEXT,plmn_nc TEXT,network TEXT NOT NULL
                            ,"\
                        "rssi TEXT NOT NULL,rssp_rscp TEXT,sinr_ecno TEXT,rsrq
                            TEXT,reg_mode TEXT,"\
                        "reg_stat TEXT,lac TEXT,cellid TEXT,zero TEXT,six TEXT,
                            lte_band TEXT,"\
                        "earfcn_dl TEXT,freq_dl TEXT,bw_dl TEXT,earfcn_ul TEXT,
                            freq_ul TEXT,"\
                        "bw_ul TEXT,aggr_num TEXT)")
        except lite.Error, e:
            log.error('Creating SQLite exception caught %s' %(e))
            sys.exit(1)

    def insertIntoSQL(s,lc):
        log.debug('Inserting transaction-id [%s] into the DB' %(s.transactionId
            ))
        cur = lc.cursor()
        try:
            cur.execute("INSERT OR REPLACE INTO Data VALUES
                (?,?,?,?,?,?,?,?,?,?,?,"\
                        "?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)",\
                        str(s.all).translate(None, "[] '").split(','))

           # s.recId = cur.lastrowid
        except lite.Error, e:
            log.error('SQLite exception caught %s' %(e))
            sys.exit(1)
        log.debug('inserted %d' %(s.recId))


    @staticmethod
    def fromHttpPost(p_data):
        log.debug(p_data)
        obj = None
        for key, value in p_data.iteritems():
            log.debug("%s=%s" % (key, value))
            obj = dataObject()
            #obj.transactionId = p_data['transaction-id'][0]
            obj.transactionId = p_data['transaction-id'][0]
            #obj.time         = p_data['timestamp'][0]
            obj.values       = str(p_data['data'][0])
            obj.all          = obj.transactionId + ',' + obj.values
        return obj


class Pi2RequestHandler(BaseHTTPServer.BaseHTTPRequestHandler):
        def do_HEAD(s):
                log.debug('Got HEAD request')
                s.send_response(200)
```

```
                s.send_header("Content-type", "text/html")
                s.end_headers()
        def do_GET(s):
                """Respond to a GET request."""
                log.debug('Got GET request')
                s.send_response(400)
                s.send_header("Content-type", "text/plain")
                s.end_headers()
                s.wfile.write("Invalid request")
        def do_POST(s):
                global LC
                """Respond to a POST request."""
                log.debug('Got POST request')
                length = int(s.headers['Content-Length'])

                dataObj = dataObject.fromHttpPost(urlparse.parse_qs(s.rfile.read
                    (length).\
                          decode('utf-8')))

                log.debug(dataObj)

                if dataObj is not None:
                    dataObj.insertIntoSQL(LC)
                    LC.commit()
                    s.send_response(200)
                else:
                    s.send_response(400)

                s.send_header("Content-type", "text/plain")
                s.end_headers()

def usage():
    print('%s usage :\n' % (os.path.basename(sys.argv[0])))
    print sys.exit(__doc__)

def main():
    global LC
    debug = False
    verbose = False
    logFile = ''
    db = '/var/www/html/modem-database.sqlite'

    listenPort = 8005
    listenAddress = '0.0.0.0'
    try:
        opts, args = getopt.getopt(sys.argv[1:],
                                   "",
                                   ["help",\
                                    "verbose",\
                                    "debug",\
                                    "db="\
                                    "log-file=",\
                                    "port=",\
                                    "address="])

    except getopt.GetoptError, err:
        print str(err) # will print something like "option -a not recognized"
        usage()
        sys.exit(2)
```

```
    for o, a in opts:
        if o in ("--debug"):
            debug = True
        elif o in ("--verbose"):
            verbose = True
        elif o in ("--db"):
            db = a
        elif o in ("--log-file"):
            logFile = a
        elif o in ("--port"):
            listenPort = a
        elif o in ("--address"):
            listenAddress = a
        else:
            assert False, "unhandled option"

    if verbose:
        log.setLevel(logging.INFO)

    if debug:
        log.setLevel(logging.DEBUG)

    if len(logFile) != 0 :
        fh = logging.FileHandler(logFile)
        fh.setLevel(logging.DEBUG)
        fh_formatter = logging.Formatter("%(asctime)s - %(name)s - %(levelname)
            s - %(message)s")
        fh.setFormatter(fh_formatter)

    log.info('Initializing monkey with SQLite(%s)' %(lite.sqlite_version))

    try:
        log.debug('Connecting db: %s' %(db))
        LC = lite.connect(db)
    except lite.Error, e:
        log.error('SQLite exception caught %s' %(e))
        sys.exit(1)

    log.debug('Create Database structure')
    dataObject.createSQLTable(LC)

    log.debug('Starting server %s:%d' %(listenAddress,listenPort))

    server = BaseHTTPServer.HTTPServer((listenAddress,listenPort),
                                       Pi2RequestHandler)

    try:
            server.serve_forever()
    except KeyboardInterrupt as e:
            sys.stdout.write('Keyboard interrupt\n')
    finally:
            server.shutdown()
            server.server_close()
            LC.close()

if __name__ == '__main__':
    main()
```

Výpis 26: Kompletní výpis souboru modem-server.py

F Zdrojový kód souboru modem-aggregator.py

Zdrojový kód je dostupný také v datové příloze této práce, v souboru `modem-aggregator.py`.

```
#!/usr/bin/env python2
import sqlite3
import time
import csv
from math import radians,sin,cos,asin,sqrt


db = "/var/www/html/modem-database.sqlite"
data_base = {}
aggr_base = {}


def mean(nums):
  return float(sum(nums))/max(len(nums),1)

def haversine(lon1, lat1, lon2, lat2):
  lon1, lat1, lon2, lat2 = map(radians, [lon1, lat1, lon2, lat2])
  dlon = lon2 - lon1
  dlat = lat2 - lat1
  a = sin(dlat/2)**2 + cos(lat1) * cos(lat2) * sin(dlon/2)**2
  c = 2 * asin(sqrt(a))
  r = 6371 # Radius of earth in kilometers. Use 3956 for miles
  return c * r

def dbCommit():
  while True:
    try:
      conn.commit()
      return
    except:
      print "dbCommit error"
      time.sleep(0.05)


conn = sqlite3.connect(db)
cur = conn.cursor()
cur.execute("CREATE TABLE IF NOT EXISTS Aggr(timestamp TEXT NOT NULL,device
    TEXT NOT NULL,"\
            "lat TEXT NOT NULL,lon TEXT NOT NULL,alt TEXT,speed TEXT,epx TEXT,
               epy TEXT,"\
            "epv TEXT,eps TEXT,csq TEXT NOT NULL,ber TEXT,plmn_cc TEXT,plmn_nc
               TEXT,"\
            "network TEXT NOT NULL,rssi TEXT NOT NULL,rsrp_rscp TEXT,sinr_ecno
               TEXT,"\
            "rsrq TEXT,reg_mode TEXT,reg_stat TEXT,lac TEXT,cellid TEXT,zero
               TEXT,six TEXT,"\
            "lte_band TEXT,earfcn_dl TEXT,freq_dl TEXT,bw_dl TEXT,earfcn_ul TEXT
               ,freq_ul TEXT,"\
            "bw_ul TEXT,aggr_num TEXT)")

newloop = 0

while True:
  cur.execute("SELECT COUNT(*) FROM Data WHERE aggr_num=0")
```

```
pending_aggregation = cur.fetchone()[0]
if pending_aggregation < 1:
  print "Records pending aggregation:", pending_aggregation
  print "Sleeping...30s"
  time.sleep(30)
else:
  print "Records pending aggregation:", pending_aggregation

  if bool(aggr_base) == False:
    for row in cur.execute("SELECT rowid,* FROM Aggr"):
      aggr_base[row[0]] = row[1:]

  for row in cur.execute("SELECT rowid,* FROM Data WHERE aggr_num like ?
      limit 10000", (0,)):
    data_base[row[0]] = row[1:]

  for dkey in data_base.keys():     # For ALL in Data
    newloop = 0
    for akey in aggr_base.keys():   # For ALL in Aggr
      if (round(haversine(float(data_base[dkey][2]),float(data_base[dkey][3])
          ,\
          float(aggr_base[akey][2]),float(aggr_base[akey][3])), 2) < 0.03 and\
          data_base[dkey][1]==aggr_base[akey][1] and data_base[dkey][14]==
              aggr_base[akey][14] and\
          data_base[dkey][22]==aggr_base[akey][22]):
        if aggr_base[akey][32] == None or int(aggr_base[akey][32]) < 1:
          num = 1
        else:
          num = int(aggr_base[akey][32])+1
        #print "num", num
        newrow = [akey] + list(data_base[dkey][0:25]) + [None,None,None,None,
            None,None,None,num]
                                                    # Create new record for
                                                        Aggr
        to_be_mean = [3,4,5,6,7,8,9,10,11,12,16,17,18,19]
        for pos in to_be_mean:
          try:
            if (pos == 3 or pos == 4):
              newrow[pos] = round(mean([float(data_base[dkey][pos-1]),\
                            float(aggr_base[akey][pos-1])]),7)
            elif (pos == 5):
              newrow[pos] = round(mean([float(data_base[dkey][pos-1]),\
                            float(aggr_base[akey][pos-1])]),1)
            elif (pos == 6):
              newrow[pos] = round(mean([float(data_base[dkey][pos-1]),\
                            float(aggr_base[akey][pos-1])]),6)
            elif (pos in range(7,11)):
              newrow[pos] = round(mean([float(data_base[dkey][pos-1]),\
                            float(aggr_base[akey][pos-1])]),2)
            else:
              newrow[pos] = int(round(mean([float(data_base[dkey][pos-1]),\
                            float(aggr_base[akey][pos-1])]),0))
          except (TypeError, NameError, ValueError) as e:
            #print "Error:", e
            newrow[pos] = data_base[dkey][pos-1]

        # print "NR:", newrow
        # print "Ndata:", [dkey] + list(data_base[dkey][:-1]) + [akey]
         aggr_base[akey] = newrow[1:]       # Save new Aggr recors
```

```
      data_base[dkey] = list(data_base[dkey][:-1]) + [akey]
      newloop = 1
      break

  if newloop == 1:
    newloop = 0
    continue
  #print aggr_base.keys()
  if bool(aggr_base.keys()) == False:
    newkey = 1
  else:
    newkey = max([int(x) for x in aggr_base.keys()])+1
  #print "aggr2", [newkey] + list(data_base[dkey][:-1]) + [1]
  #print "data2", [dkey] + list(data_base[dkey][:-1]) + [newkey]
  aggr_base[newkey] = list(data_base[dkey][:-1]) + [1]
  data_base[dkey] = list(data_base[dkey][:-1]) + [newkey]

# WRITE DOWN ALL CHANGES
try:
  cur.execute("DROP TABLE AggrTemp")
except Exception as e:
  print e
  #quit()
cur.execute("CREATE TABLE IF NOT EXISTS AggrTemp(timestamp TEXT NOT NULL,
    device TEXT NOT NULL,"\
            "lat TEXT NOT NULL,lon TEXT NOT NULL,alt TEXT,speed TEXT,epx
               TEXT,epy TEXT,"\
            "epv TEXT,eps TEXT,csq TEXT NOT NULL,ber TEXT,plmn_cc TEXT,
               plmn_nc TEXT,"\
            "network TEXT NOT NULL,rssi TEXT NOT NULL,rsrp_rscp TEXT,
               sinr_ecno TEXT,"\
            "rsrq TEXT,reg_mode TEXT,reg_stat TEXT,lac TEXT,cellid TEXT,zero
                TEXT,six TEXT,"\
            "lte_band TEXT,earfcn_dl TEXT,freq_dl TEXT,bw_dl TEXT,earfcn_ul
               TEXT,freq_ul TEXT,"\
            "bw_ul TEXT,aggr_num TEXT)")

print "aggr1", aggr_base[1]
for akey in aggr_base.keys():
  newrow = [akey] + list(aggr_base[akey])
  cur.execute("INSERT OR REPLACE INTO AggrTemp(rowid,timestamp,device,lat,
     lon,alt,speed,"\
             "epx,epy,epv,eps,csq,ber,plmn_cc,plmn_nc,network,rssi,
                rsrp_rscp,"\
             "sinr_ecno,rsrq,reg_mode,reg_stat,lac,cellid,zero,six,lte_band
                ,"\
             "earfcn_dl,freq_dl,bw_dl,earfcn_ul,freq_ul,bw_ul,aggr_num)"\
             "VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,"\
             "?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)", newrow)
try:
  cur.execute("DROP TABLE Aggr")
except Exception as e:
  print e
  #quit()
cur.execute("ALTER TABLE AggrTemp RENAME TO Aggr")
#print "NumOfKeysInDataBase:", len(data_base.keys())
for dkey in data_base.keys():
  #print 'dkey:', dkey
```

```
  cur.execute("UPDATE Data SET aggr_num=? WHERE rowid=?", [data_base[dkey
     ][32],dkey])
dbCommit()      # Save to disc
data_base = {}
```

Výpis 27: Kompletní výpis souboru modem-aggregator.py

G Zdrojový kód webové stránky index.php

Zdrojový kód je dostupný pouze v datové příloze této práce, v souboru `index.php`.